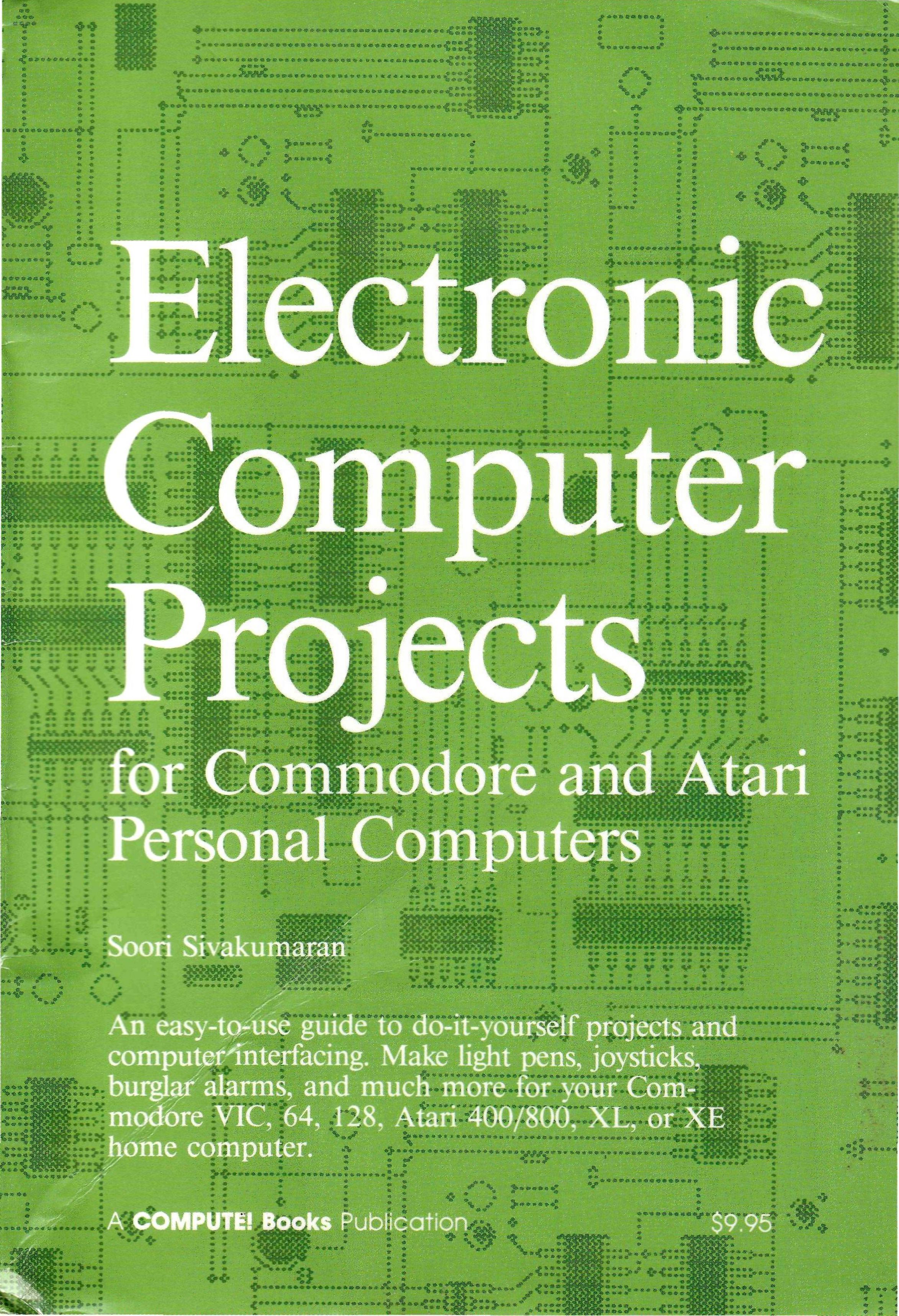


Electronic Computer Projects

for Commodore and Atari Personal Computers

Soori Sivakumaran

An easy-to-use guide to do-it-yourself projects and computer interfacing. Make light pens, joysticks, burglar alarms, and much more for your Commodore VIC, 64, 128, Atari 400/800, XL, or XE home computer.

A **COMPUTE! Books** Publication

$9.95

VISIT…

LANZAROTE
Caliente.com

Electronic Computer Projects

for Commodore and Atari Personal Computers

Soori Sivakumaran

COMPUTE!™ Publications,Inc. abc
Part of ABC Consumer Magazines, Inc.
One of the ABC Publishing Companies
Greensboro, North Carolina

Copyright 1986, COMPUTE! Publications, Inc. All rights reserved.

Reproduction or translation of any part of this work beyond that permitted by Sections 107 and 108 of the United States Copyright Act without the permission of the copyright owner is unlawful.

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

ISBN 0-87455-052-1

The author and publisher have made every effort in the preparation of this book to insure the accuracy of the programs and information. However, the information and programs in this book are sold without warranty, either express or implied. Neither the author nor COMPUTE! Publications, Inc., will be liable for any damages caused or alleged to be caused directly, indirectly, incidentally, or consequentially by the programs, information, and/or projects in this book; nor liable for any accidents or errors in the implementation of the projects, the handling of any materials for the projects, and for any damage that may result to any individual or equipment.

The opinions expressed in this book are solely those of the author and are not necessarily those of COMPUTE! Publications, Inc.

COMPUTE! Publications, Inc., Post Office Box 5406, Greensboro, NC 27403, (919) 275-9809, is part of ABC Consumer Magazines, Inc., one of the ABC Publishing Companies, and is not associated with any manufacturer of personal computers. Atari, Atari 400, Atari 800, Atari XE, and Atari XL are trademarks of Atari Corporation. Commodore 64, Commodore 128, and VIC-20 are trademarks of Commodore Electronics Limited. Radio Shack is a trademark of Tandy Corporation.

Contents

Foreword v

Preface vii

1 Getting Inside Your Computer 1
2 Exploring the Control Port 7
3 Making a Joystick 21
4 Game Paddles 31
5 An Analog Light Sensor 43
6 A Light Pen 51
7 A Digital Light Sensor 65
8 An Electronic Switch 79
9 A Burglar Alarm 95
10 Digital Logic 107
11 A Better Logic Probe 121
12 More Ideas 127
13 Robotics 149

Appendices 163
A COMPUTE!'s Guide to Typing In Programs 165
B Integrated Circuits 171

Index 181

Foreword

Your personal computer is a powerful machine. It can calculate faster than any clerk, print documents faster than any typist, and play chess with the masters. Your software library is probably full of such applications, along with games, utilities, and educational software.

But a computer can do a lot more than run software. It can be a sophisticated controller which turns lights on and off, monitors your home's windows and doors, and makes robot limbs move. To accomplish these tasks, however, you need sensors, motor circuits, and other electronic devices. That's why you need *Electronic Computer Projects.*

This book is a step-by-step guide to building a variety of electronic devices, from the simple to the sophisticated. With complete and concise instructions anyone can follow, and accompanied by detailed photographs and figures, *Electronic Computer Projects* is the book with which you can teach your Commodore or Atari personal computer valuable new tricks.

Parts lists outline what you need, the instructions show you how to do everything from heating the soldering iron to plugging in the last component, and programs test and adjust each project.

Learn how to build your own joysticks, game paddles, and light pens. See how to put together two kinds of light sensors, even how to create a burglar alarm with an infrared sensor that "sees" in the dark. Logic probes, which let you "look" into digital circuits, multiplexers and demultiplexers, even robotic motor control circuits are all part of this book's intriguing and educational projects. Each project has been built and tested—you won't find any surprises when you connect a device to your computer.

Electronic Computer Projects is a thorough guide to your adventure into computer control of the outside world.

Preface

Computers are versatile machines which can be programmed to perform tasks quickly and accurately. These tasks, or *applications,* run the range from games and word processing to financial modeling or controlling the lights and heat in your home.

Most computer applications, at some point in the process, involve a program which processes *data,* or information. A video arcade game is a good example. Here, the data is entered by the player through a joystick—the processing the program performs is the motion of the character on the screen. Another example, word processing, lets you enter text (the data) and process it (by formatting it properly and printing it out). These kinds of applications are certainly useful, but they aren't the end of your computer's capabilities.

A vast majority of computer applications are limited to acting only on data entered by the user, either from the keyboard or the joystick. Yet some of the most interesting applications result when computers interact with the outside world. A computer which can examine its surroundings can then use this information to perform a useful function. Imagine your computer, sensing that it's now dark outside, turning on your outside lights before you arrive home from work.

Computers have made possible systems which can control extremely complex operations. The *software,* or programs which instruct a computer to do a specific task, is in fact a form of intelligence—if written correctly, the programs can react to different conditions in predetermined ways.

This book will show ways you can use your computer to relate and react to the outside world. This is done through *sensors* and *actuators.* Sensors gather information from the outside world for the computer to process. Actuators allow the computer to influence outside events. Several of the projects put sensors and actuators together so that the computer can perform a given task.

To help you get started, the first few projects in this book are relatively easy to complete. As your experience increases, so does the difficulty of the projects. If you're new to electronics, then, start with the first projects in the book, developing your skills as you progress toward the later, more difficult, projects. But even though the projects increase in difficulty, you won't need any exotic equipment or advanced skills to complete them.

Almost all the projects contain less than a half-dozen components. You'll have little difficulty with any of the projects, but you should have some understanding of electronics troubleshooting procedures.

Building a Circuit

The basic idea when building a circuit is to wire together the components' leads correctly. Occasionally, even experienced builders make mistakes in wiring circuits. It's an excellent idea to completely read through each section describing the circuit's construction *before* starting. Color-coded wires can help in tracing circuits. Examine the drawing of the breadboard layout included in the book, and study the schematic diagram. When you feel you understand what the circuit requires, go ahead and wire your test circuit.

Several methods of wiring, or *breadboarding*, circuits are available and suitable for the projects in this book. One method involves mounting the components on a perforated board. The components are positioned on one side of the board so that their leads extend through the perforations to the opposite side. The leads on the bottom are then connected by wires soldered between them.

A similar method, called *wire wrapping*, requires that each component be mounted in a socket before being placed on the perf board. Each socket has metal posts which make contact with the leads of the component. The posts extend through the holes. Wire connections are made by wrapping one end of a wire around one post and the other end of the same wire around a second post. This is done with a wire wrap tool.

Most commercial electronic circuits, however, are constructed using printed circuit boards. Printed circuit boards are copper plated on one or both sides. When making a circuit using printed circuit boards, you must first determine the location of components on the board. Then, the interconnections between the components' leads are marked on the board with an indelible felt-tip marker. The board is submersed in a chemical solution which dissolves all copper from the board except that along the traces you've marked. The copper traces which remain form the connections between the components. Commercial manufacturers prefer this technique because boards can be produced in large quantities quickly and cheaply with photographic techniques.

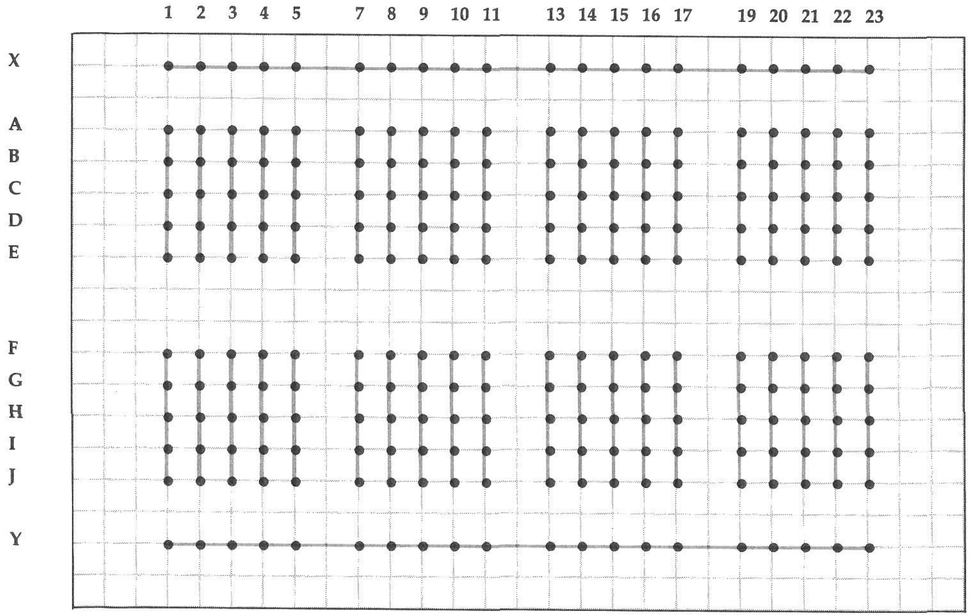
Your best bet is to use solderless breadboards, like those available from Radio Shack. (Radio Shack's Experimenter Socket, part number 276-174, was used to construct all the projects in this book.)

A solderless breadboard is a plastic board with a grid of "holes" called *plug points* (see Figure 1). The plug points are internally wired so that the columns of points on each half of the board are connected. A row of plug points on each half of the board are connected as well. To hook up a circuit, components are just plugged into the solderless breadboard. A connection between two component leads can be made simply by inserting each lead into plug points that are internally connected. No soldering is required. Alternatively, jumper wires can be used to bridge one set of connected plug points with another to complete a connection. Jumper wires should be #22 gauge, as larger wires tend to spread the contacts of a plug point too far apart, eventually ruining the breadboard. These solderless boards may be reused for different experiments without removing solder from connections. Most wiring explanations in this book refer to coordinates on the solderless breadboard.

Heat the Iron

Even if you use a solderless breadboard when constructing the circuits in this book, you'll still have to do a *little* soldering. While you won't have to do intricate soldering, wires still must be attached to switches, the connectors which plug into your computer, and other components that cannot be directly

Figure 1. Pin Diagram of a Solderless Breadboard


1
2
3
4
5
7
8
9
10
11
13
14
15
16
17
19
20
21
22
23
X
A
B
C
D
E
F
G
H
I
J
Y

Top view of a solderless breadboard. Colored lines indicate internal connections. Dots mark plug points.

plugged into a solderless breadboard. Soldering is the only reliable method of making these connections.

To make solder connections, you need a pencil soldering iron (approximately 25 watts) and some electronic resin core solder. Make sure the solder you use is suitable for electronic work—some types of solder used in other applications use a corrosive resin. The steps to make a solder connection are listed below. Try soldering some practice connections with scraps of wire before soldering on one of the projects.

1. Make sure the joint you're going to solder is free of dirt and grease. The solder will not bond properly if it's not clean.
2. Try to make the joint as firm as possible before soldering. If you're joining two wires, twist them together. When connecting to a terminal, like that of a switch, twist the wire around it. Doing this makes the connection more reliaible, as a dab of solder by itself will not hold a connection together for long. If you're using stranded wire, twist the strands together before making the connection. This will prevent stray strands from shorting out nearby connections.
3. When you're soldering an electronic component such as a transistor or IC (integrated circuit) chip, attach an alligator clip to the lead of the component just above the point where you're going to make the connection. The metal alligator clip acts as a heat sink, drawing heat away from the component, preventing damage.
4. Plug in the soldering iron and wait until its tip is hot enough to melt solder.
5. Touch the tip of the soldering iron to the joint and wait a couple of seconds for it to heat up.
6. Touch the solder to the joint, *not* the tip of the soldering iron. Remember always to heat the work surface, *not* the solder. Some solder should flow onto the joint. Remove the remaining solder from the joint and then remove the soldering iron. A minimum amount of solder should be used when making a connection. Excess solder not only looks sloppy, but can cause short circuits.

7. A good solder connection is bright and shiny. If it's a dull gray, you have what's called a *cold joint.* If this happens, remake the connection. A poor solder joint can cause an otherwise perfect circuit to fail.
8. Sometimes it will be necessary for you to *tin,* or coat, the leads to a component with a thin coating of solder before establishing a connection. Tinning helps remove oxidation and cleans the service, providing a better electrical and mechanical connection.
9. When you're finished soldering, wipe the tip of your soldering iron on a damp sponge to remove any excess solder and resin. The tip should be a silver color, without any trace of dirt or foreign objects. This practice will increase the useful life of the tip. Be sure to unplug the soldering iron as well—an unattended hot iron is a fire hazard.

In Your Toolbox

Besides a soldering iron, there are a few other tools you'll need. A pair of small wire cutters and a wire stripper are handy, but you can strip insulation from a wire with a knife if you're careful not to nick the conductor. Needle-nose pliers are useful for twisting and untwisting wire, and for reaching tight spots too small for your fingers. A drill and a screwdriver are required for some of the projects to mount the circuits in a case.

Since you'll be working with integrated circuits (ICs), an IC extractor will be handy for removing the IC chip from the circuit board. You'll find that the pins of the ICs are very easy to bend.

If the circuit doesn't give the results you expect after you've carefully built it, compare your circuit to the diagram in the book. Are all the pins properly connected? Is the IC inserted correctly? Have you entered the program designed to use the circuit properly? Most programs included with this book are short, so you should have little trouble entering them correctly. To make this part of the job even easier, use "The Automatic Proofreader" found in Appendix A—it's an error-checking program which insures that you type in the program correctly the first time.

Each circuit described in this book has been thoroughly tested. The circuits and the demonstration programs will work as explained in the text, providing you follow the instructions carefully.

All projects are designed to operate with the Commodore 64, Commodore 128 (in 64 mode), and VIC-20, and with the Atari 400, 800, 600XL, 800XL, and 130XE personal computers. The electronics involved is virtually identical for all these computers, but the programs differ. Be sure to use the correct program for your computer.

Have fun building the projects. Experiment with your own ideas. By the time you've finished the last project, you should understand how to interface your computer to almost anything. At that point, your own imagination will take control.

Help's on the Way

To make it easier for you to use this book, you'll see small graphic devices, called *icons*, throughout the book. These cues will alert you to specific sections of each chapter:

shows you where the parts list for each project is located. The number of components, their names, and their Radio Shack part numbers are provided.

points out where the step-by-step instructions for building each project begin. Steps are numbered and are self-explanatory for the most part.

indicates that a testing procedure is described, or that a program will follow. After you've finished a project, you'll almost always be shown ways to test it under working conditions to insure that it operates as advertised.

calls your attention to various warnings and/or notes on a project.

CHAPTER 1

Getting Inside Your Computer

1 Getting Inside Your Computer

A computer is really nothing more than a series of switches. Of course, it's a highly complex series of switches. Thousands and thousands of them. Like any switches, those you find in a computer can have two positions, on or off. Storing information in a computer is simply a matter of setting a sequence of switches on or off. If the switch is *on,* its value is *1.* If the switch is *off,* it holds a value of *0.*

The *digital circuits* used in computers are switches which perform a specified series of actions, all based on the information in the on and off states of the circuit. Everything in a digital circuit is either on or off, unlike *analog circuits,* found in things like radios and televisions, which deal with voltage levels and signals. Analog circuits may amplify or otherwise modify electronic signals of various levels.

These on and off states within a computer are represented as electronic signals of two different voltage levels. In the computers this book deals with, an *on* state corresponds to a signal of about +5 volts, while the *off* state corresponds with a signal of about 0 volts. (Actually, a positive voltage more than 2.5 volts is considered *on,* and a voltage less than 0.5 volts is considered *off.* This is called a *TTL,* or transistor-transistor logic signal level.)

Since a switch can exist in only two states, on or off, what information can be represented? Simple—one of two numbers. The on state can be assigned to mean the number 1, and the off state can be the number 0—the two digits which make up the base two binary number system. The binary number system naturally lends itself for use with the digital logic circuitry in a computer. Unfortunately, we're used to counting by ten. The number 65, for instance, has far more meaning to us than its binary equivalent, 1000001.

Dealing with numbers at the binary level is best left to the computer, but when you're trying to interface your computer to the outside world, it's often necessary to know and understand how setting a bit in a memory address within the computer affects the machine and its operation.

Number Systems

The binary number system is just another way of representing numbers. The number system we normally use, the decimal number system, is a base ten number system. In other words, there are ten different symbols (0, 1, 2, ..., 9) which can be used independently or in combination to represent a given number. The value of the number represented depends on both the combination of symbols and their relative positions. Let's take a look at how a decimal number is made up, for example 37,506.

Position **43210**
Decimal number **37506**

Symbol		**Base**	**Position**		**Decimal Value**
3	*	10	^4	=	30,000
7	*	10	^3	=	7,000
5	*	10	^2	=	500
0	*	10	^1	=	0
6	*	10	^0	=	6
					37,506

As you can see, the first symbol, 3, represents three times ten to the fourth power (in other words, a 3 followed by four zeros). In the same manner, each symbol represents a number multiplied by a power of ten (the power is determined by the symbol's relative position). Adding all these values together gives us the number 37,506.

The binary number system has just two symbols, 0 and 1. A BInary digiT, or *bit*, can thus be either 0 or 1. One bit by itself can only be used to count up to one. Larger numbers in binary, as in the decimal number where you can represent larger value numbers with more digits (a number like 245 requires three digits, whereas the number 4 requires just one), are represented by more bits. As you can see in the following example, a binary number is made in a similar manner as a decimal number.

Position	**76543210**
Binary number	**11111001**

Symbol		**Base**	**Position**		**Decimal Value**
1	*	2	^7	=	128
1	*	2	^6	=	64
1	*	2	^5	=	32
1	*	2	^4	=	16
1	*	2	^3	=	8
0	*	2	^2	=	0
0	*	2	^1	=	0
1	*	2	^0	=	1
					249

Just as with the more comfortable decimal numbers, binary numbers are built by combining symbols, then multiplying those values by two (since binary is base two) raised to the power of the symbol's position. A 1 in the seventh position, then, represents the decimal value 128, while a 1 in the first position only represents 1.

Here are a few more examples of how decimal numbers (what we use) are represented as binary numbers (what computers use). Try working them out to see if you come up with the same values.

Decimal		Binary
10	=	1010
5	=	101
134	=	10000110

Chock Full of Bits

In computers, bits are usually put together in groups of eight when stored or otherwise manipulated. A group of eight bits is called a *byte*. Since it has eight positions, a byte can hold any number from 00000000 binary (0 decimal) to 11111111 binary (255 decimal). The largest number which can be represented by a byte, then, is 255.

These binary values are used by the computer for all its operations, from executing an applications program to representing alphanumeric, even graphics characters.

Understanding the elements of the binary number system, bits, and bytes will prove useful as you move into the heart of *Electronic Computer Projects*. The next chapter, "Exploring the Control Port," is your start toward your own custom-built computer projects.

CHAPTER 2

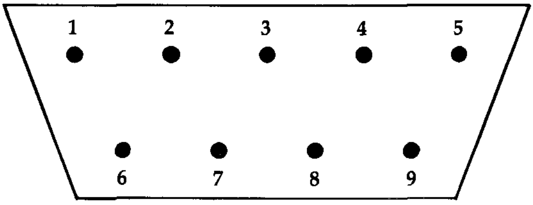
Exploring the Control Port

2 Exploring the Control Port

With a basic understanding of how a computer operates, you can now begin exploring the computer's ports, the gateways through which information flows into and out of your computer. The ports are often referred to as *I/O* ports, which stands for *input/output* ports. Though there are several different I/O ports on your computer, the projects in this book use one port in particular, the joystick port.

The joystick port provides plenty of access to your computer for our demonstation circuits. Connectors for this port are also commonly available. They are the same type as those found on joysticks and game paddles—D-subminiature female connectors.

Figure 2-1. The Joystick Port



Control Port 1		**Control Port 2** (Commodore 64)	
Pin	**Type**	**Pin**	**Type**
1	JOYA0	1	JOYB0
2	JOYA1	2	JOYB1
3	JOYA2	3	JOYB2
4	JOYA3	4	JOYB3
5	POT AY	5	POT BY
6	Button A/LP	6	Button B
7	+5 volts	7	+5 volts
8	Ground	8	Ground
9	POT AX	9	POT BX

The joystick port has nine pins. Pins 1, 2, 3, 4, and 6 are I/O pins. They are connected to wires which may be used to communicate binary (on/off) data. Pins 5 and 9 are used to input analog data—we'll learn more about these pins later in the book. Pin 7 is connected to a +5-volt source, while pin 8 is connected to the power supply ground.

Once you can identify the pins of a control port, your next step is to understand how they can be used. A few homemade "tools" will help.

A Simple Logic Probe

A logic probe is a device that lets you "look" into what's going on in a digital circuit. It's a device that tells you whether a 1 (high state) or a 0 (low state) is present at a point in the circuit. A simple logic probe can be assembled from two main components. This logic probe can be used to see how the I/O lines of the control port are used to input information.

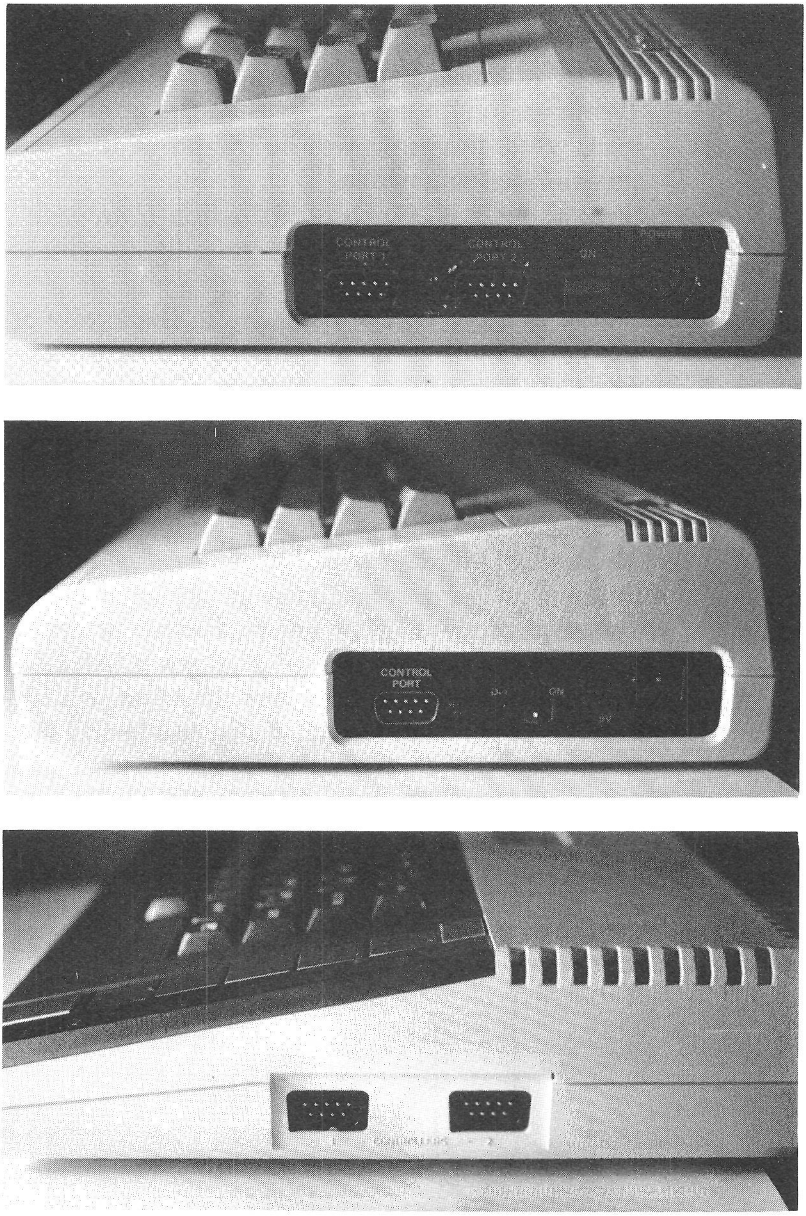
To construct the logic probe, you'll need these parts:

Quantity	**Part**	**Part Number**
1	Light-emitting diode (LED)	276-041
1	1K ohm resistor	271-8027
2	Alligator clips	270-378

You'll also need some stranded copper wire (about #22 gauge—Radio Shack part number 278-1307) to make the probe leads, as well as some electrical insulation tape.

(In this book, all part numbers are Radio Shack part numbers.)

Figure 2-2. Commodore 64, VIC-20, and Atari Joystick Ports


CONTROL PORT 1
CONTROL PORT 2
ON
POWER
CONTROL PORT
ON
9V
1
CONTROLLERS
2

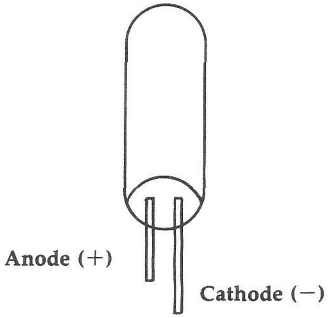
The joystick ports for the Commodore 64 (top), Commodore VIC-20 (middle), and Atari 800XL (bottom) are shown in these photos. Note that the VIC-20 has only one joystick port.

Here's the procedure for putting together your simple logic probe:

1. Cut two pieces of stranded copper wire about 10 inches long. One wire will be the ground lead, while the other will be the probe lead of this simple logic probe.
2. Strip about a half inch of insulation from each end of both wires and attach an alligator clip to one end of each wire.
3. Solder the free end of one wire to the anode of the light-emitting diode (LED). Usually, the anode leg of an LED is the shorter of the two legs, but this may vary with manufacturers. The wire connected to the LED's anode is the *probe* lead of the logic probe.

Note: An LED lights up when current passes through it in the proper direction, which is from anode to cathode. LEDs come in various colors, but the most common are red and green. An LED will conduct current in only one direction. It's important that you don't confuse the anode and the cathode leads. In fact, if an LED fails to work in a circuit, the first thing to check is whether it's been installed correctly (see Figure 2-3). The cathode lead is usually the longer of the two leads.

Figure 2-3. A Typical LED



4. Solder the cathode of the LED to one end of the 1K ohm resistor.
5. Solder the other end of the 1K ohm resistor (brown, black, and red bands) to the free end of the second wire. This second wire is now the ground wire of the logic probe.
6. To complete your simple logic probe, cover your soldered connections with some electrical insulation tape. This will prevent short circuits from occurring while you use your probe.

Note: *Resistors* limit the flow of electricity in a circuit and are measured in ohms. The value of a resistor is indicated by its colored stripes—usually there are four. The first two stripes' colors represent numbers as determined by the resistor color code (see Table 2-1). These numbers are the first two digits of the resistor value. The third stripe is the *multiplier.* The number corresponding to the multiplier color is the power of ten by which the first two numbers are multiplied (an easy way to remember it is simply to add x number of zeros to the first two numbers, where x is the multiplier color's value). All three numbers are combined to give the value (in ohms) of the resistor.

Thus, a resistor which has as its first three stripes (as the one you're working on has)

Brown
Black
Red

will have a resistance of 1000 ohms. The brown stripe provides the 1, the black stripe gives the first 0, and the red stripe provides the next two 0's (10×10^2).

The fourth stripe represents the resistor's *tolerance*. The tolerance gives the range, plus or minus, which the actual value of the resistor will be within when compared to the strict value indicated by the color bands. The color of this stripe can be gold, silver, or salmon. The tolerances are plus or minus 5, 10, and 20 percent respectively. Sometimes you won't find a tolerance stripe on a resistor. This means that the resistor has a tolerance of plus or minus 20 percent. If you have a resistor with brown, black, red, and silver stripes, for example, you have a 1000 ohm (1K ohm) resistor with a 10 percent tolerance. The actual value of the resistor is thus between 900 and 1100 ohms. The circuits in this book can all use resistors with a 20 percent tolerance or better.

Table 2-1. Resistor Color Codes

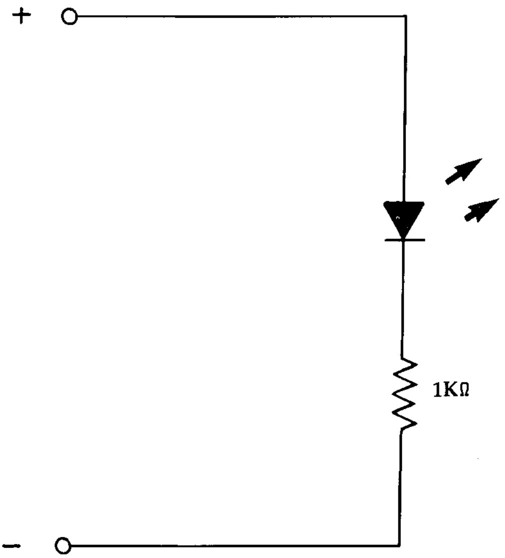
Color	**Value**
Black	0
Brown	1
Red	2
Orange	3
Yellow	4
Green	5
Blue	6
Violet	7
Gray	8
White	9

Tolerance Bands

Color	**Tolerance**
Gold	± 5 percent
Silver	± 10 percent
Salmon	± 20 percent
None	± 20 percent

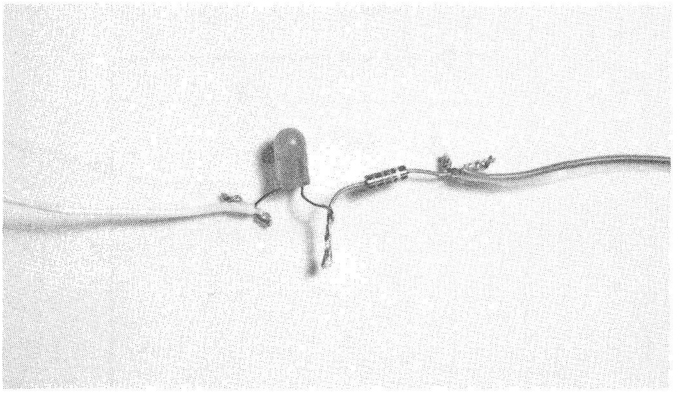
Resistors also have power ratings, which describe the amount of energy (measured in watts) which the resistor can safely handle. For the projects in this book, you may use 1/4- or 1/2-watt resistors. A resistor, unlike many other circuit components, may be connected in either direction in a circuit.

Figure 2-4. Schematic Diagram of a Simple Logic Probe



Using the simple logic probe is very easy. Attach the ground lead of the logic probe to the power supply ground of the digital circuit you're testing. Then simply touch the tip of the probe lead to a point in the circuit. If the point has a voltage of something greater than the barrier voltage of the LED, usually about 0.7 volts, the LED will glow. You can assume the point is at a logic *high* state (a 1). If the point is at a voltage level less than 0.7 volts, current will not flow through the diode, the LED stays off, and you can assume a logic *low* state (a 0) is present. (Of course, a zero indicates a voltage level less than 0.5 volts, so a problem could still exist with the circuit under test and not be detected by this simple tester.)

Figure 2-5. The Completed Logic Probe



When you're finished building the simple logic probe, it should look something like this.

To use the logic probe to examine the joystick port, a connector is required—the next simple tool.

A Connector for the Joystick Port

The actual connector that plugs into the joystick port is a D-subminiature female, 9-pin connector. Stranded copper wire (#22 gauge) should be soldered to the terminals of the connector for easier access.

Here's what you'll need:

Quantity	**Part**	**Part Number**
1	9-pin D-subminiature female	276-1538

You'll also need some stranded copper wire (about 12 feet) or ribbon cable.

The procedure is quite simple.

1. Cut six pieces of stranded copper wire about 24 inches long and strip about 1/4 inch of insulation from each end. (You may wish to use ribbon cable instead of wire to avoid a tangle.)
2. Solder a wire to the terminals of pins 1, 2, 3, 4, 6, and 8 of the 9-pin plug.
3. Tape a tag to the opposite end of each wire with the number of the pin it's attached to. This will save you some time later.

To use your logic probe with the connector, attach the ground lead of the probe to the wire from pin 8.

While your computer is turned off, insert the plug into the joystick port. (You should always make sure that the computer is turned off whenever you insert a plug into any of the I/O ports.) If you're using a Commodore 64 or Commodore 128, insert the plug into joystick port 2 on the right side of the machine. If you have an Atari computer, insert the plug into joystick port 1.

Warning: Before turning on your computer, make sure that none of the wires touch each other. If any wires are touching, your computer could be damaged. Be particularly careful to inspect the wires from pins 7 and 8—they should never be in contact. If they are, the computer's power supply could be damaged.

Touch the end of your logic probe to pins 1, 2, 3, 4, and 6. Each time you do this, the LED should glow. This glow will be very faint, and you may need to shield the LED from room light to detect the glow at all. This is because of the small amount of current available at the pins.

The I/O lines are connected to registers, or memory addresses, in your computer. The five pins (1, 2, 3, 4, and 6) of joystick port 2 correspond to the lower five bits of memory location 56320 ($DC00 in hexadecimal) in the Commodore 64 and Commodore 128. Whether you're using the port for input or output is determined by the data direction register located at 56322 ($DC02). Each bit controls the direction of data on the corresponding bit of the port. If a bit is set to 1, the corresponding bit in the port is set to be used for data *output*. Setting the bit to 0 causes the bit to be used for *input*. In the VIC-20, these lines correspond to bits in memory locations 37137 ($9111) and 37154 ($9122). In the Atari computers, you needn't worry about the memory locations affected, as the BASIC STICK and STRIG commands can be used to see the effect of the five I/O lines.

The bits in these registers are normally set *on* (1) in your computer—that's why the LED of your logic probe should glow faintly when you touch the probe to any of these pins. But how can these pins be used to input information?

Type in and run the appropriate version of Program 2-1.

Program 2-1—Commodore 64/128

```
EX 10 A=56320:REM FOR PORT 1 USE 56321
FQ 20 IFPEEK(A)AND(2↑0)THEN40
PH 30 PRINT"PIN 1"
RM 40 IFPEEK(A)AND(2↑1)THEN60
SK 50 PRINT"PIN 2"
MG 60 IFPEEK(A)AND(2↑2)THEN80
BP 70 PRINT"PIN 3"
SQ 80 IFPEEK(A)AND(2↑3)THEN100
ER 90 PRINT"PIN 4"
AD 100 IFPEEK(A)AND(2↑4)THEN120
SJ 110 PRINT"PIN 6"
SM 120 GOTO20
```

Program 2-1—VIC 20

```
AH 10 A=37137
BS 20 IFPEEK(A)AND(2↑2)THEN40
PH 30 PRINT"PIN 1"
GF 40 IFPEEK(A)AND(2↑3)THEN60
SK 50 PRINT"PIN 2"
SA 60 IFPEEK(A)AND(2↑4)THEN80
BP 70 PRINT"PIN 3"
BX 80 POKE 37154,0:IFPEEK(37152)AND(2↑7)
      THEN100
ER 90 PRINT"PIN 4"
XE 100 POKE 37154,255:IFPEEK(A)AND(2↑5)T
       HEN120
SJ 110 PRINT"PIN 6"
SM 120 GOTO20
```

Program 2-1—Atari

```
FP 10 IF STICK(0)<>14 THEN 20
EP 15 PRINT "PIN 1"
GA 20 IF STICK(0)<>13 THEN 30
FB 25 PRINT "PIN 2"
GA 30 IF STICK(0)<>11 THEN 40
FD 35 PRINT "PIN 3"
DH 40 IF STICK(0)<>7 THEN 50
FF 45 PRINT "PIN 4"
DN 50 IF STRIG(0)<>0 THEN 60
FI 55 PRINT "PIN 6"
AA 60 GOTO 10
```

When you run the program for your computer, nothing should seem to happen at first. Take your logic probe and touch the probe to the wire connected to pin 8. This is the ground from the computer and should still be connected to the ground lead of the probe. Obviously, the ground lead and probe lead are at the same potential. The LED will not light. Since ground voltage corresponds with the *off* state, everything is working correctly.

Now, take the wire from pin 1 and touch its free end to the wire from pin 8. When you do this, you force the normally *on* pin 1 to turn *off.* Your program tells you this. If you look at the computer screen, the message, PIN 1 should be displayed on the screen. As you touch the wires from pin 2, 3, 4, or 6 to the ground, you'll see a message showing which pin has been set *low.* By forcing an I/O pin to logic *low,* its corresponding bit in the register (a memory location) is set low as well. Since a program can PEEK into a memory location and check its contents, software can be written to detect the states (on or off) present at the control port, then proceed accordingly.

In Program 2-1, your computer is sensing which pin you're setting low (by touching the wire from the pin to ground). Then it's taking this input and processing it; in this case it's used to print out messages. The next chapter will use this information to construct a useful input device, a joystick.

CHAPTER 3

Making a Joystick

3 Making a Joystick

It may seem that a joystick, with its molded plastic parts, would be impossible to make at home. Open a Commodore or Atari joystick, though, and you'll see that it's not that complex. In fact, it only contains five switches.

Four of the joystick's switches are used to detect the direction the control stick is pushed. The fifth acts as a fire button. A switch is pressed if the joystick is moved up, down, left, or right. If the stick is moved diagonally, *two* switches are pressed. Moving the stick diagonally up and to the right, for example, presses both the "up" and "right" switches.

These switches are used to make the connection between one of the five input lines (pins 1, 2, 3, 4, and 6) of the control port and the ground (pin 8). Instead of your touching wires together, as you did while experimenting with the control port, switches are closed to make the connections.

Since a joystick is basically a set of switches, you can make your own version of this device. In your first joystick, four push button switches are substituted for the control stick. The four buttons correspond to the directions a joystick can indicate—up, down, left, and right. To get diagonal directions, you'll press two buttons, the top and right buttons to indicate a move upward and toward the right, for instance. The fire button is a fifth push button switch and functions just as does its counterpart on a commerical joystick.

As you can imagine, the push button joystick works better with some programs than with others. The push button joystick lets you move with greater control in games and in applications where movement in the four main directions is required. Applications or games requiring diagonal moves, however, are trickier than with a standard joystick.

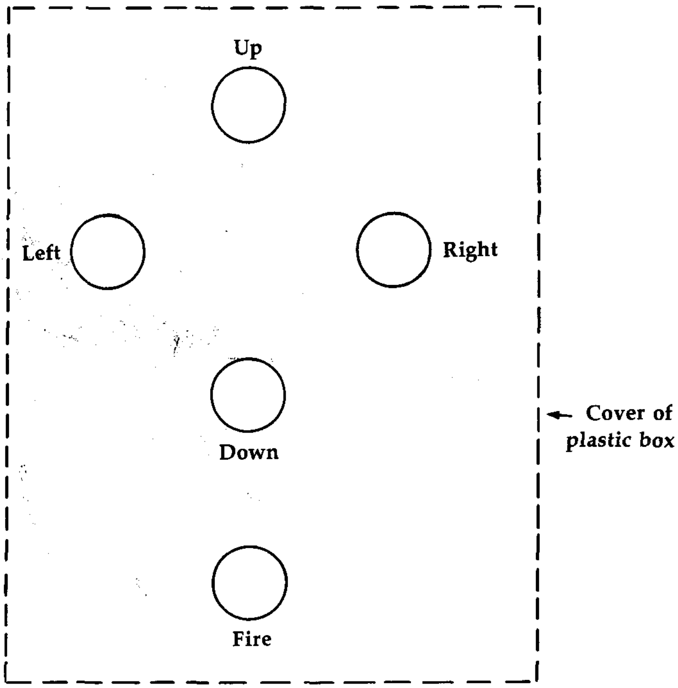
To make a push button joystick for your Commodore or Atari computer, you'll need the following parts:

Quantity	Part	Part Number
5	Momentary contact normally open switches	275-1547
1	Plastic case	270-231
1	9-pin D-subminiature female	276-1538

You'll also need some stranded copper wire or ribbon cable.

1. Cut six pieces of wire about 24 inches long. Remove about 1/4 inch of insulation from each end. Solder wires to pins 1, 2, 3, 4, 6, and 8 of the 9-pin plug.
2. Remove the cover from the plastic case and drill five holes for the switches (Figure 3-1).
3. Mount a switch in each of these holes.
4. Drill a hole in the top end (the end closest to the up switch) of the plastic case for the wires from the connector.
5. Pull the six wires from the 9-pin plug through the hole you just drilled, and about 10 centimeters from their ends, tie them in a knot so that they can't be pulled back through the hole.
6. Solder the wire from pin 8 to one of the terminals of a switch (probably the up switch, since it's closest to the hole you drilled in the top side of the cover). Using several small lengths of wire, connect this terminal to a terminal of each of the other four switches in a jumper fashion (Figure 3-2).

Figure 3-1. Drilling the Cover

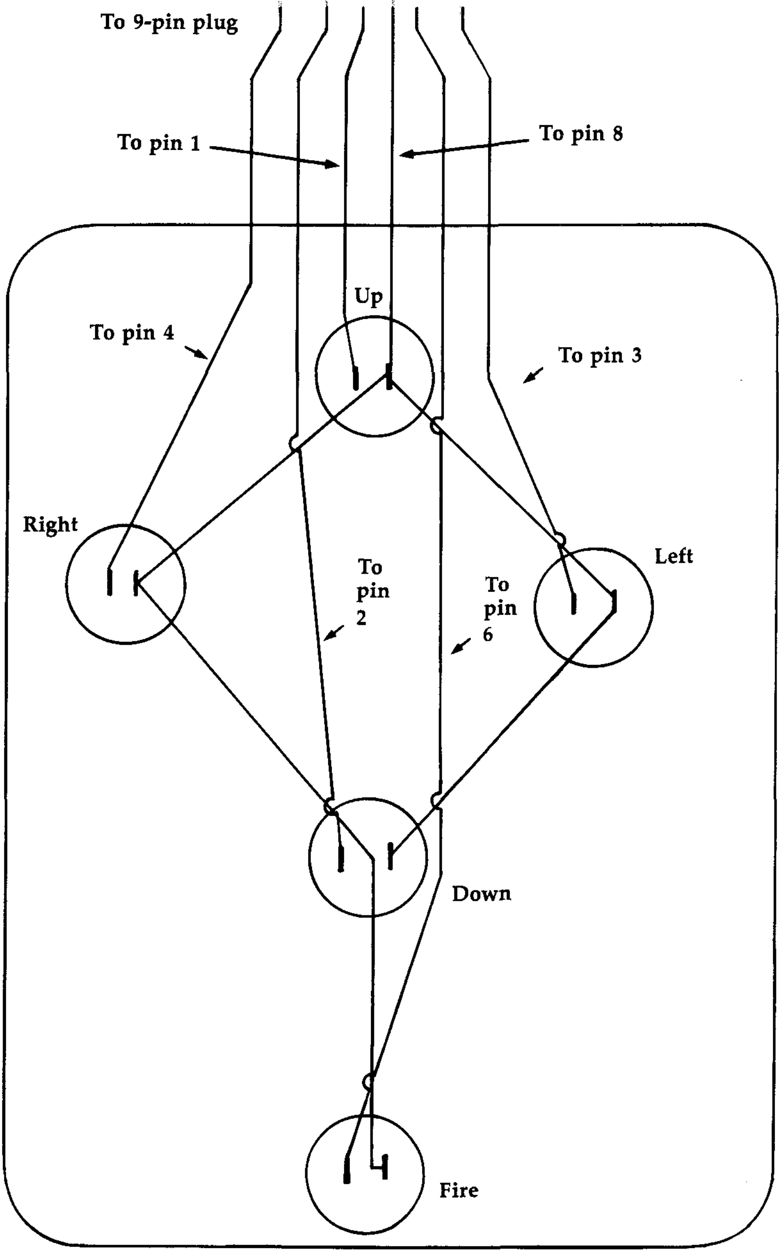


This front view of the plastic case shows the approximate positions where you should drill the five holes for the joystick's button switches.

7. Take the wire from pin 1 of the plug and solder it to the free terminal of the up switch (see Figure 3-2 for a layout view, or Figure 3-3 for a schematic view).
8. Solder the wire from pin 2 to the free terminal of the down switch.
9. Solder the wire from pins 3, 4, and 6 to the free terminals of the left, right, and fire switches respectively.
10. Replace the cover of the plastic case. Your joystick is complete.

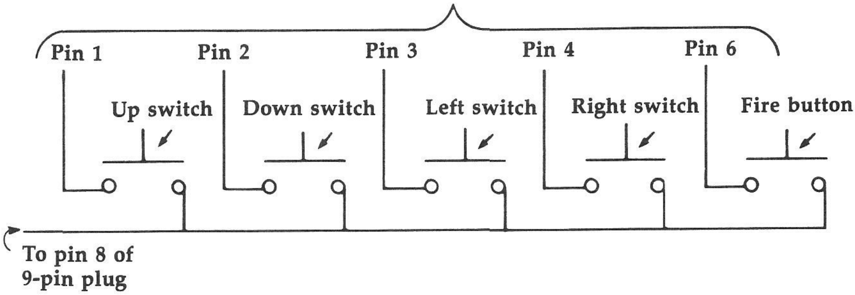
4 2 1 8 6 3

Figure 3-2. Connecting the Switches


To 9-pin plug
To pin 1
To pin 8
Up
To pin 4
To pin 3
Right
To pin 2
To pin 6
Left
Down
Fire

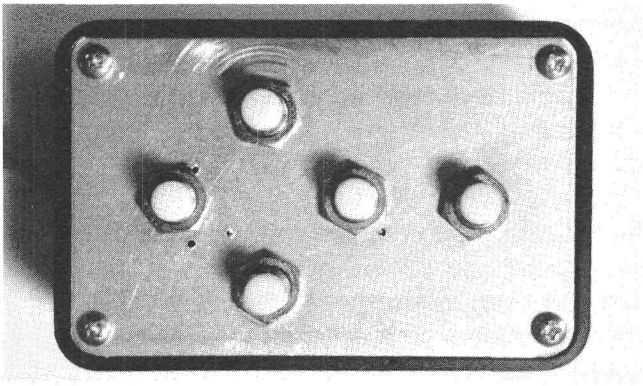
In this back view of the joystick cover, you can see how the switches are wired. Notice that the wire from pin 8 of the 9-pin connector is first connected to the up switch, then shorter pieces of wire are used to connect a terminal on each of the other four switches.

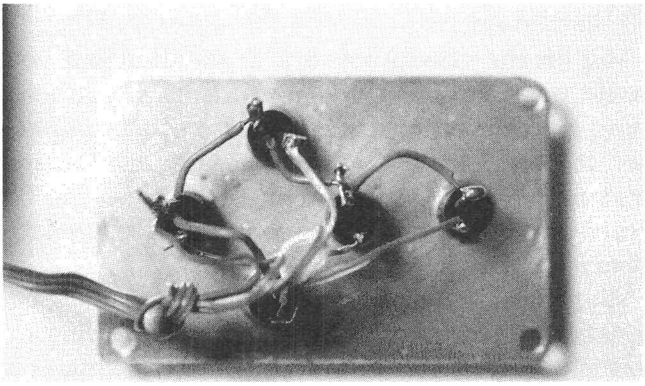
Figure 3-3. Joystick Schematic


Pin 1
Pin 2
Pin 3
Pin 4
Pin 6
Up switch
Down switch
Left switch
Right switch
Fire button
To pin 8 of
9-pin plug

Schematic view of the push button joystick wiring.

Figure 3-4. The Finished Push Button Joystick





The completed push button joystick looks like this. Five buttons appear on the top of the finished case (above), while the wiring can be clearly seen in the view from the bottom (below).

A Gravity Joystick

You can also use other types of switches to make a joystick. One alternative is to use *tilt* switches in place of the four directional push button switches.

A tilt switch is a switch which closes only when it's tilted in the proper direction. One type of tilt switch consists of a plastic case with a metal ball inside. When the case is held at the proper angle, gravity causes the ball to roll to one side, where it touches both terminals of the switch, closing the contact. Some tilt switches use liquid mercury in place of the metal ball.

If you're using tilt switches, be sure to mount them inside the plastic case of your joystick to prevent them from breaking.

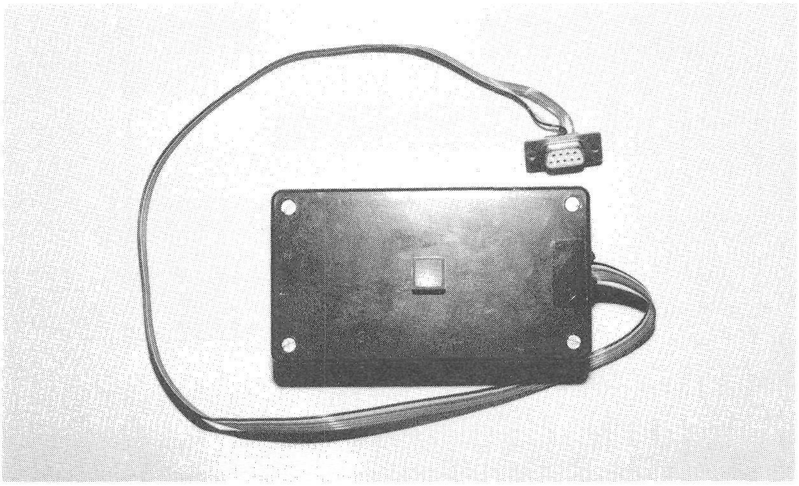
Warning: This is especially important if you're using mercury switches. Mercury is a potential health hazard.

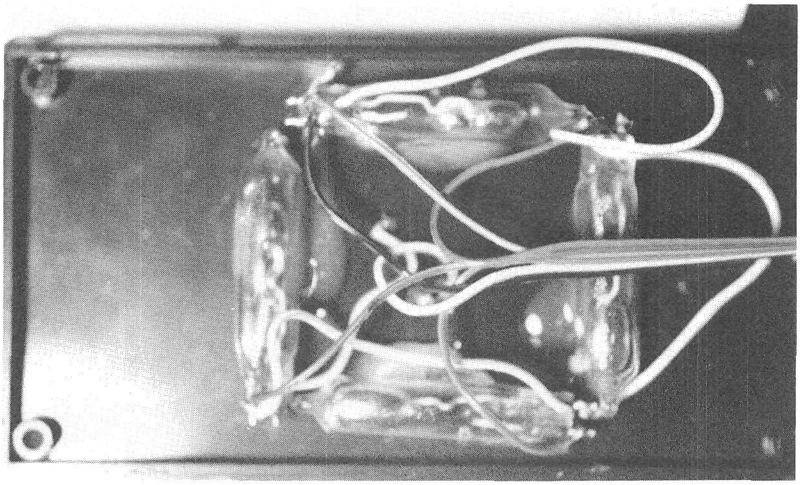
The switches can be glued to the inside of the case using epoxy and should be mounted so that they close (turn on) when the joystick is tilted in the proper direction. This means the right switch should close when the case is tilted to the right, the left switch should close when the case is tilted left, and so on.

You tilt the gravity joystick in the direction you would normally push the control stick if you were using an ordinary joystick. Because of its unique feel, the gravity joystick adds another dimension to game playing. While moving objects on the screen is easy, keeping them stationary requires steady hands.

Your joystick, whether push button or tilt switch, will work with any programs requiring a joystick. In fact, you may want to try making your own custom controller, perhaps something as unusual as a steering wheel for a racing game.

Figure 3-5. Inside a Gravity Joystick





The gravity joystick has only one button (the fire button), since it's operated by tilting it in the proper direction (above). Tilting the joystick case closes contacts within one or more of the mercury switches (below).

CHAPTER 4

Game Paddles

4 Game Paddles

Game paddles can make arcade-style games like car racing simulations, tennis, and Ping-Pong more realistic and fun. Unlike joysticks, game paddles are *analog* devices. While joysticks provide the computer with either *on* or *off* signals, paddles provide the computer with a varying signal.

A paddle contains a resistor with a variable amount of resistance and a switch in one case. The switch acts as the fire button and operates like the fire button on a joystick. The electrical resistance of the paddle depends on the position of the variable resistor, often referred to as a *potentiometer,* or *pot.* When the pot is turned all the way in one direction, the resistance measured between the potentiometer's side and middle leads is 0 ohms. When the shaft is turned in the opposite direction, the resistance is about 500K ohms on Commodore paddles. Atari paddles require a resistance of about 1M (1 million) ohms.

When paddles are used, current from the computer flows through the potentiometers. The amount of current that flows depends on the resistance in the pot. The greater the resistance, the less current flow. The opposite is also true—the less the resistance, the greater the current flow.

By turning the knob of the potentiometer, the voltage, measured across the leads of the potentiometer, varies between 0 volts (ground) and +5 volts. The computer senses this voltage and uses it to determine a value for the position of the potentiometer.

This voltage, which is an analog value, must be converted to a digital value to be used by your computer. *Analog-to-digital converters,* sometimes called *A/D converters,* were once very expensive.

Fortunately, your computer has this hardware built in. Using its analog-to-digital converter, your computer interprets the voltage across the variable resistor of the paddle as a number between 0 and 255 (1 to 228 for the Atari). This value is kept in special memory locations in your computer. For the Commodore 64 and 128, these memory locations are 54297 and 54298 ($D419 and $D41A in hexadecimal). In the VIC-20, locations 36872 and 36873 ($9008 and $9009) are used. For the Atari computers, it's not necessary to PEEK the actual memory locations—Atari BASIC has a command, PADDLE(X), which lets you see the values returned for the paddle positions. The value of X in this Atari statement is 0 or 1 (a different X for each paddle), provided the paddles are plugged into control port 1.

To build your own pair of game paddles, you'll need the following parts:

Quantity	**Part**	**Part Number**
2	Plastic cases	270-231
2	Momentary contact switches	275-1566
1	9-pin D-subminiature female	276-1538
2	Plastic knobs	274-407
2	500K potentiometers (Commodore)	271-210
or		
2	1M potentiometers (Atari)	271-211

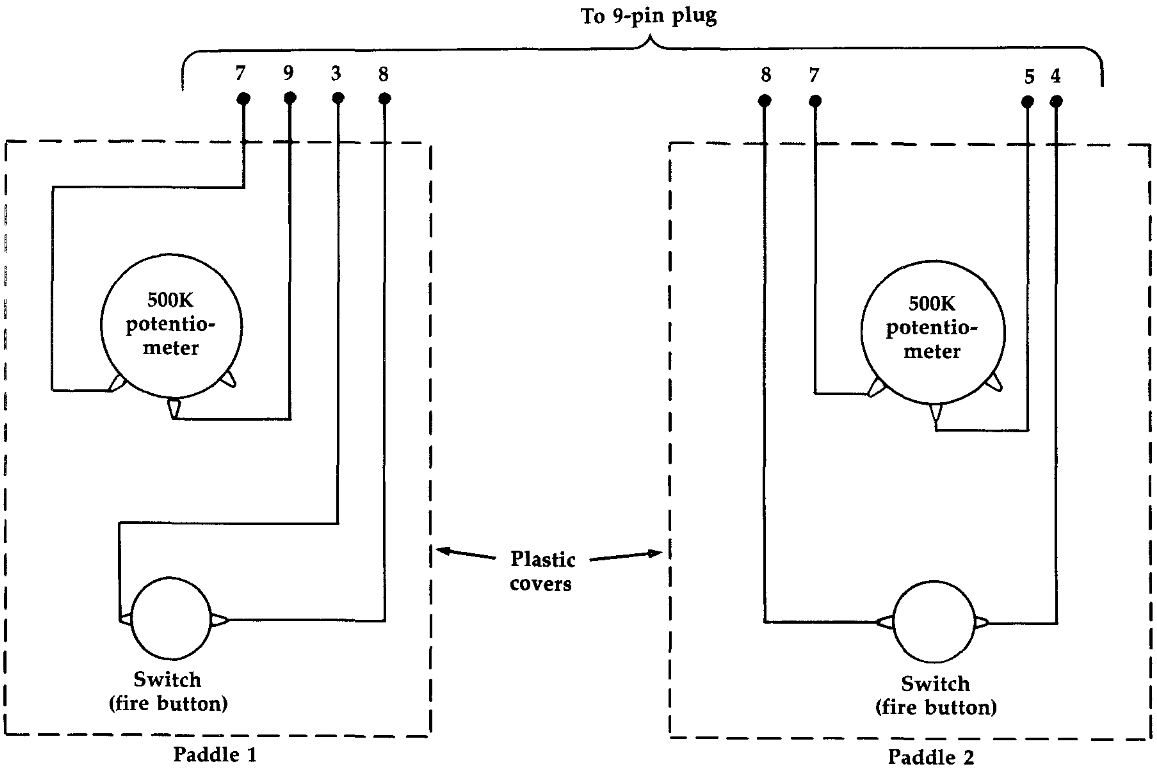
You'll also need some ribbon cable or stranded copper wire to connect your paddles to your computer.

Here's how to put your game paddles together:

1. Cut eight pieces of stranded copper wire about 24 inches long and remove about 1/4 inch of insulation from each end.

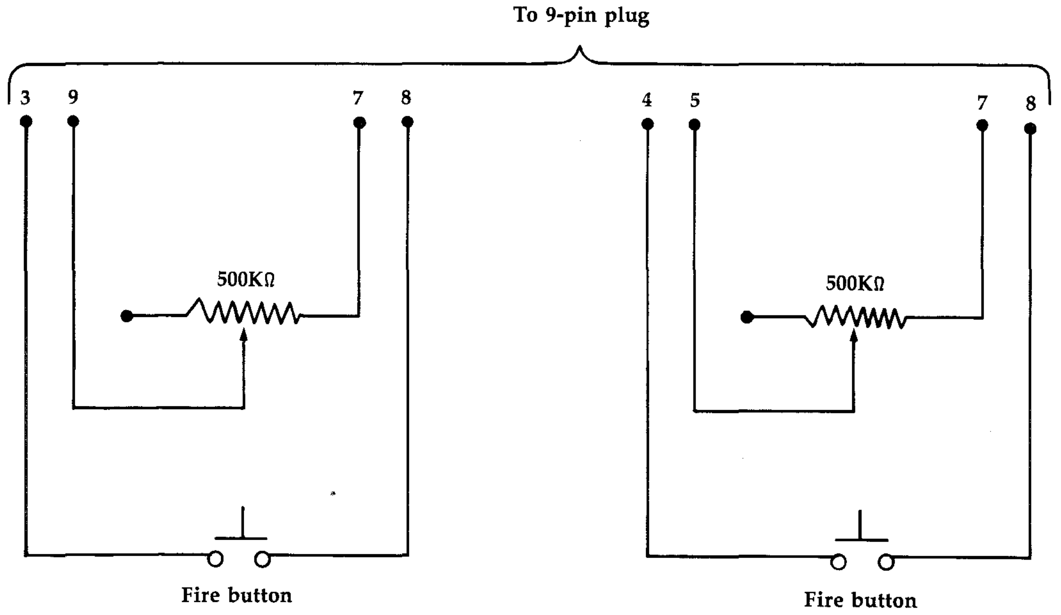
2. Connect one end of two of the wires to pin 7 of the 9-pin plug.
3. Connect another two wires to pin 8 of the plug.
4. Solder the remaining four wires to pins 3, 4, 5, and 9.
5. Remove the cover from one of the plastic boxes and drill two holes as in Figure 4-1. One hole is for the potentiometer, while the other is for the switch. Drill a hole in one end of the plastic case for the wires. Repeat these steps for the second plastic box.
6. Mount the switches and potentiometers in their respective holes. Attach the knobs to the shafts of the potentiometers.
7. Take one wire from pin 7, one wire from pin 8, the wire from pin 3, and the wire from pin 9 of the 9-pin plug to one of the paddle boxes. Pull the wires through the hole in the top end of the box. Turn the box cover with the potentiometer and switch mounted on it over so that their leads are facing up (as shown in Figure 4-1).
8. Connect the wire from pin 8 to one lead of the switch. Solder the wire from pin 3 to the other switch lead. The wire from pin 7 should be soldered to the left side lead of the potentiometer, and the wire from pin 9 is connected to the middle lead of the potentiometer. Mount the cover on the box. One paddle is completed.
9. The procedure for the second paddle is identical, except the wires are from pins 7, 8, 4, and 5 of the 9-pin plug. The wire from pin 8 connects to one lead of the switch. The other switch lead is connected to the wire from pin 4. The wire from pin 7 connects to the left lead of the potentiometer, and the remaining wire, from pin 5, connects to the middle lead. Close the plastic box, and your second paddle is finished.

Figure 4-1. Game Paddles—Bottom View


To 9-pin plug
7
9
3
8
8
7
5
4
500K potentio- meter
500K potentio- meter
Plastic covers
Switch (fire button)
Switch (fire button)
Paddle 1
Paddle 2

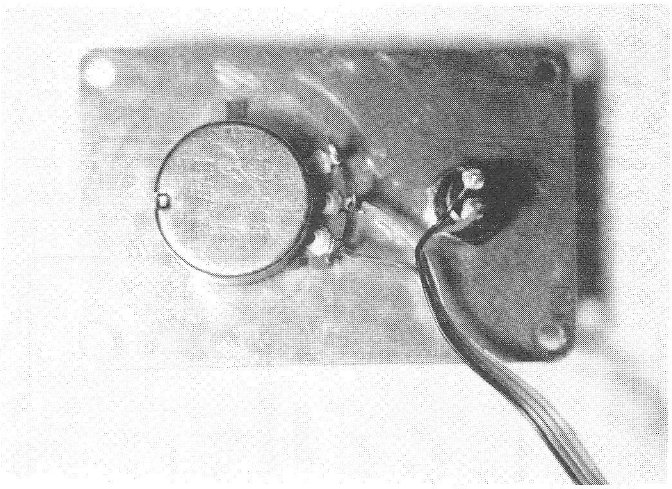
Drill holes in each plastic case as shown here for the potentiometer and the switch. Another hole must be drilled in the top end of the case for the wires which will connect the paddle to the computer.

Figure 4-2. Schematic Diagram of Game Paddles


To 9-pin plug
3
9
7
8
500KΩ
Fire button
4
5
7
8
500KΩ
Fire button

In this schematic view, you'll notice that the potentiometer leads are switched from those shown in Figure 4-1. This figure represents a top view of the potentiometer and switch.

Figure 4-3. Under the Paddle



If you've correctly drilled holes, placed components, and wired leads, the bottom view of the game paddle case should look something like this.

To try out your paddles, turn off your computer and plug the paddles' 9-pin connector into control port 1. Turn on the computer, then type in and run the appropriate version of Program 4-1.

Program 4-1—Commodore 64/128

```
EF 10 POKE56333,127:REM IRQ OFF
HH 20 POKE56322,192:REM SET BITS 6,7 OF
      {SPACE}DATA DIRECTION REGISTER FOR
       OUTPUT
EC 30 POKE56320,64:REM 128 FOR PORT 2
HG 40 X=PEEK(54297)
DJ 50 Y=PEEK(54298)
XE 60 POKE56322,255:POKE56333,129:REM RE
      STORE REGISTERS
DG 80 PRINTX,Y
SJ 100 GOTO10
```

Program 4-1—VIC-20

```
GD 10 PRINT1-(PEEK(37151)AND16)/16;PEEK(
      36872);
RG 20 POKE37154,127:PRINT1-(PEEK(37152)A
      ND128)/128;:POKE37154,255
PS 50 PRINTPEEK(36873)
BD 60 GOTO10
```

Program 4-1—Atari

```
10 PRINT PADDLE(0);" ";PADDLE(1)
20 GOTO 10
```

When you run the appropriate program, two columns of numbers should appear on the screen. Each column of numbers corresponds with a paddle. As you turn the knobs of the paddles, the numbers should vary between 0 and 255 for the Commodore computers, and between 1 and 228 for an Atari computer.

Several steps are necessary to read the paddles with a Commodore 64 computer. Since the keyboard can interfere with values read at address 56321 ($DC01), it's necessary to turn off the IRQ (Interrupt Request). This prevents normal system processing from occurring. Program 4-1 does this in line 10.

Bits 6 and 7 of the data direction register must be set up for output by setting the proper bits in address 56322 ($DC02). You want to place a 1 in bits 7 and 6 of 56322 for data direction register A. (The decimal equivalent 192 of the binary number 11000000 in line 20 sets bits 6 and 7 for output.)

Line 30 stores the value 64 in the data port register A to select reading the paddles at port 1. (To read port 2, change the 64 in line 30 to 128.)

Lines 40 and 50 PEEK the game paddle POTX and POTY locations, storing the proper values in the variables X and Y.

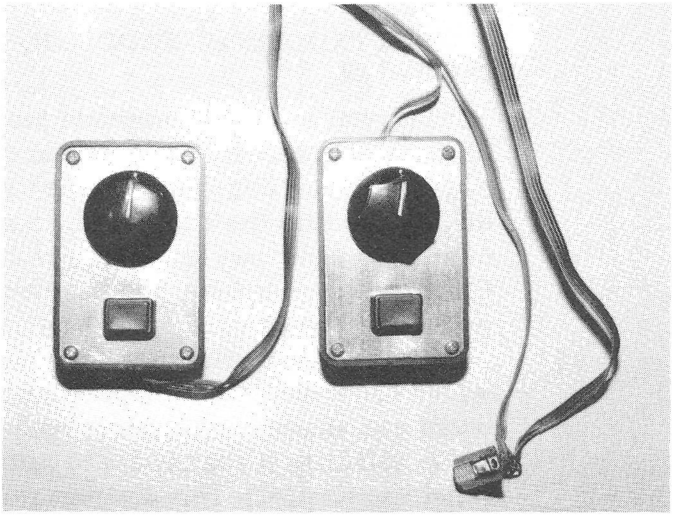
the variables X and Y.

Line 60 restores the IRQ and keyboard.

Line 80 prints the values of X and Y to the screen.

Since it's necessary to turn off the keyboard for input to the paddle registers, it's often best to read the values with a machine language (ML) routine.

Figure 4-4. Paddles Done



Both game paddles are wired to one 9-pin connector.

The following ML routine (Program 4-2) works only on the Commodore 64 or Commodore 128 in 64 mode. The routine is POKEd into the cassette buffer and can be called by typing SYS 828 anytime you want to read the paddles. It's a simple matter to incorporate this routine's BASIC loader (omit line 40) into any program you write which requires paddle reading. To read a paddle value after calling the routine with SYS 828, simply PEEK the appropriate location:

	Paddle to Read	Location to PEEK
Port 1	X	251
	Y	253
Port 2	X	252
	Y	254

Program 4-2. ML Paddle Reader

```
MF 10 I=828:AD=251
HG 20 READ A:IF A=256 THEN40
RG 30 POKE I,A:I=I+1:GOTO20
QA 40 SYS 828:PRINTPEEK(AD);PEEK(AD+1);P
      EEK(AD+2);PEEK(AD+3):GOTO40
RF 50 DATA 76,63,3,162,1,120,173
KH 60 DATA 2,220,133,167,169,192,141
SX 70 DATA 2,220,169,128,141,0,220
EE 80 DATA 160,128,234,136,16,252,173
DS 90 DATA 25,212,149,251,173,26,212
CD 100 DATA 149,253,169,64,202,16,232
FP 110 DATA 165,167,141,2,220,88,96,256
```

CHAPTER 5

An Analog Light Sensor

5 An Analog Light Sensor

Sometimes it's handy to be able to detect different intensities of light. Perhaps you'd like to have a simple switch to turn on your outside lights at dusk and then turn them off again at dawn. If you need to detect different degrees of lighting, you need an analog light sensor.

The analog light sensor built in this chapter is a cadmium sulfide photocell. It's a variable resistor whose resistance changes proportionally to the amount of light which strikes its light-sensitive surface. This, in turn, provides the computer with information regarding just how much light is present. The resistance increases as the amount of light striking it decreases. Exposed to bright light, the cadmium sulfide photocell has a resistance of a few hundred ohms between its terminals. In darkness, this resistance increases to approximately 0.5 megohms—about the same value range as the potentiometer on the Commodore paddles. In fact, it's possible to connect the cadmium photocell in much the same manner as a paddle.

Light Sensor—Commodore Computers

To build an analog light sensor for a Commodore 64, Commodore 128, or VIC-20, you'll need the following parts:

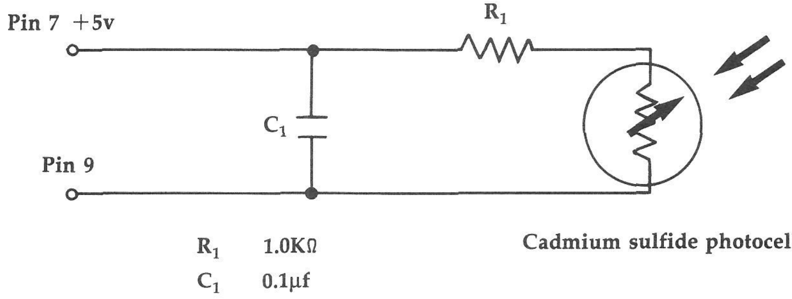
Quantity	Part	Part Number
1	Cadmium sulfide photocell	276-116
1	9-pin D-subminiature female	276-1538
1	0.1 microfarad capacitor	272-1069
1	1K ohm resistor	271-8023
1	Solderless breadboard	276-175

You'll also need some stranded copper wire, as well as some solid copper wire (Radio Shack part number 278-1306).

To complete the project, follow these steps:

1. Cut two pieces of the stranded copper wire about 24 inches long and strip about 1/4 inch of insulation from each end. Cut several small pieces of solid wire about 2 inches long to use as jumpers. (You'll need them in this circuit and most others in this book.)
2. Solder the end of one wire to pin 7, and the other wire to pin 9 of the 9-pin plug.
3. Connect the wire attached to pin 7, the +5-volt lead, to point X1 on the solderless breadboard. Place the other wire (that attached to pin 9) to point Y1 on the board.
4. Install jumper wires between points Y2 and J2, points X2 and A2, and points J13 and Y13.
5. Place the 0.1 microfarad capacitor between points F2 and E2.
6. Bend the leads of the 1K ohm resistor, and install it between points B2 and B13.
7. Install the cadmium sulfide photocell between points E13 and F13.

Figure 5-1. Schematic Drawing of Analog Photocell (Commodore)



Building an analog light sensor for a Commodore personal computer requires several parts, including a cadmium sulfide photocell, a resistor, and a capacitor.

Light Sensor—Atari

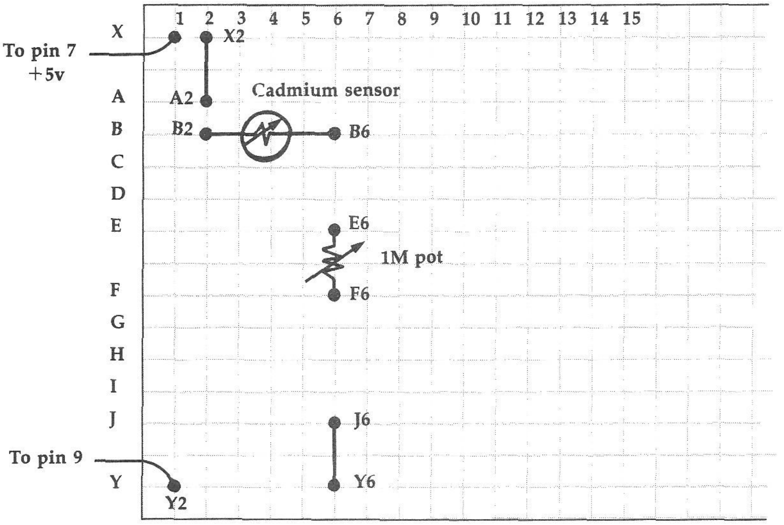
Putting together an analog light sensor for the Atari computer series involves some different components and different directions.

You'll need:

Quantity	Part	Part Number
1	Cadmium sulfide photocell	276-116
1	9-pin D-subminiature female	276-1538
1	1M potentiometer	271-211
1	Solderless breadboard	276-175

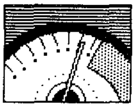
You'll also need some stranded copper wire as well as some solid copper wire.

Figure 5-2. Breadboard Layout for Atari Analog Photocell


1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
X
A
B
C
D
E
F
G
H
I
J
Y
To pin 7
+5v
X2
A2
B2
Cadmium sensor
B6
E6
1M pot
F6
J6
To pin 9
Y6
Y2

The breadboard for the Atari version of the analog light sensor clearly shows where each component and jumper should be placed.

1. Cut two pieces of the stranded copper wire about 24 inches long and strip about 1/4 inch of insulation from each end. Cut several small pieces of solid wire about 2 inches long to use as jumpers.
2. Solder the end of one wire to pin 7, and the other wire to pin 9 of the 9-pin plug.
3. Connect the wire attached to pin 7, the +5-volt lead, to point X1 on the solderless breadboard. Place the other wire (that attached to pin 9) to point Y2 on the board.
4. Install jumper wires between points Y6 and J6, and points X2 and A2.
5. Place the 1M potentiometer between points E6 and F6.
6. Install the cadmium sulfide photocell between points B2 and B6.



To see how your analog light sensor works, use Program 4-1 from the previous chapter. If you're using a Commodore 64, or a Commodore 128 in 64 mode, you can use Program 5-1 below instead. If you have an Atari computer, enter and run the two-line program you see below. For a Commodore 64 or 128, the light sensor should be plugged into port 2. Plug the Atari light sensor into port 1.

Program 5-1—Commodore 64/128

```
AD 10 I=828
PE 20 READ A:IF A=256 THEN50
RG 30 POKE I,A:I=I+1:GOTO20
FX 40 REM ANALOG LIGHT METER
GH 50 SYS 828:A=PEEK(252)
DH 60 IFB=ATHEN50
GX 70 B=A:PRINT"{CLR}LIGHT LEVEL IS NOW"
      ;A
DG 80 GOTO50
KA 90 DATA 76,63,3,162,1,120,173
FR 100 DATA 2,220,133,167,169,192,141
BJ 110 DATA 2,220,169,128,141,0,220
QG 120 DATA 160,128,234,136,16,252,173
QE 130 DATA 25,212,149,251,173,26,212
CS 140 DATA 149,253,169,64,202,16,232
FJ 150 DATA 165,167,141,2,220,88,96,256
```

Program 5-1—Atari

```
10 PRINT PADDLE(0)
20 GOTO 10
```

When the you run the program, numbers in the range 0–255 (1–228 for the Atari) will scroll on your screen. These numbers are related to the amount of light which is falling on the photocell. To see this, try reducing the light reaching the photocell by covering it with your hand. As you do, the numbers in the column will increase. Conversely, increasing the amount of light which hits the photocell, perhaps by shining a flashlight on it or by holding it closer to a lamp, causes the numbers on the screen to decrease.

Note: On the Commodore light sensor, you may need to adjust the value of R1, the 1K ohm resistor, to cause a swing through the entire range of values. In other words, you may need to use a different resistor, one rated higher or lower than 1K. Just experiment until you find the proper value for your application. That's part of the fun in breadboarding—making changes is easy.

The computer treats the light sensor as though it were a game paddle. The varying resistance of the sensor provides the computer with a signal between 0 volts and +5 volts at its port. The voltage is converted to a digit between 0 and 255 by the analog-to-digital converter in your computer. This value is stored in a memory location, where it can be read by a program. It shouldn't be hard, once you have the sensor built, to take this information and develop a light meter program which will determine the degree of light present at the sensor.

Two analog light sensors can be connected to your computer through the single 9-pin plug (just like two game paddles). The second photocell is attached between pins 7 and 5 of the 9-pin plug. To read the value of this second sensor, simply PEEK memory location 54298 ($D41A) for the Commodore 64/128, or PEEK location 36873 ($9009) for the VIC. Don't forget to set the data direction registers for output. The BASIC command PADDLE(1) can be used to read this value for Atari computers.

CHAPTER 6

A Light Pen

6

A Light Pen

A light pen is a fascinating input device. By pointing a light pen at the computer's screen and pressing its input button, you can make selections from a menu, draw a picture, or perform whatever function the computer program you're using allows. The biggest attraction of a program which uses a light pen is that you don't have to use the keyboard—using a program becomes as simple as pointing.

To understand how a light pen works, you must know a bit about how your computer's monitor displays a picture. Pictures on the screen are made up of tiny dots of light. These dots are produced from an electron gun in the monitor which shoots electrons at the screen. These electrons excite the phosphor coating on the screen, causing it to emit light. In this way, the electron gun actually "paints" the picture, dot by dot, row by row, until the screen is filled. All of this is done in a fraction of a second and is updated many times every second.

Your computer keeps track of where the beam of electrons from the electron gun is at all times. The light pen, when pointed to a spot on the screen, triggers the computer to store the location of the beam as it passes. These x and y coordinates are placed in special registers of your computer. Since the position of the beam stored is the same as that of the light pen, programs can determine where you're pointing the light pen by checking the values contained in these registers. The program can then use this information to perform various functions, like drawing a line or selecting an option.

Here's what you'll need to make a simple light pen:

Quantity	Part	Part Number
1	Solderless breadboard	276-175
1	7400 quad 2-input NAND gate IC	276-1801
1	TIL infrared phototransistor	276-145
1	1K ohm resistor (Commodore only)	271-8023
1	9-pin D-subminiature female	276-1538
1	Push button microswitch (normally open)	275-016

You'll also need some stranded copper wire, a ballpoint pen, and some solid copper wire.

When building the light pen, refer to Figure 6-1, the schematic drawing for the light pen circuit, and Figure 6-2, the solderless breadboard layout, as you follow these instructions:

1. Solder lengths of solid copper wire to pins 6, 7, and 8 of the 9-pin plug. These wires will connect the plug with the solderless breadboard.
2. Insert the wire from pin 6 to point E1 of the breadboard, the wire from pin 7 to point Y1, and the wire from pin 8 to point X1.
3. Insert the 7400 IC in the breadboard so that its pin 1 is at point E16 and its pin 8 at point F10. You can tell which pin of the IC is pin 1 by one of two methods: Pin 1 may have a circle indented in the plastic beside it. Alternatively, there may be a semicircular indention at the end of the IC where pin 1 is located. In this case, pin 1 is the first pin in a counterclockwise direction from this indention (looking at the top of the IC). In either case, the remain-

ing pins are numbered counterclockwise around the IC.

4. Insert five solid copper jumper wires as follows:

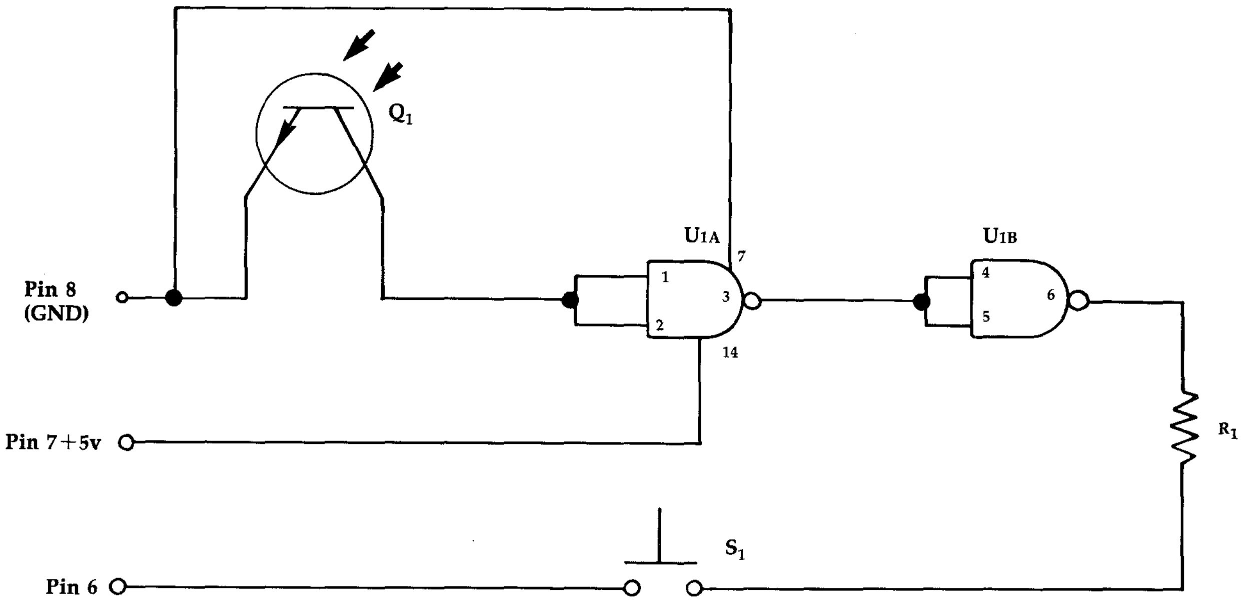
Wire	**First Location**	**Second Location**
1	A10	X10
2	A12	A13
3	C13	D14
4	C15	D16
5	J16	Y16

5. Remove the ink cartridge from the ballpoint pen. The barrel of the pen will act as the body of your light pen. With your hot soldering iron, carefully melt a small hole at the end of the barrel where the ballpoint used to be for the wires.

Warning: Be sure to clean off the tip of your soldering iron when you're finished. If you intend to use your soldering iron for serious work, it's a good idea to change the tip of your iron after using it to melt plastic. Substances from the burning plastic will adhere to the tip of your iron, making it unsuitable for serious soldering work. [We found that it wasn't necessary to burn a hole for installing the TIL 414 phototransistor when we used a BIC ballpoint pen—Editor.]

6. Cut four pieces of stranded copper wire about 30 inches long to connect your light pen with the solderless breadboard. These wires must be long enough to allow you to move the light pen freely.
7. Solder one of the four wires to the emitter lead of the phototransistor. Another wire should be soldered to its collector lead. Attach a small piece of electrical insulation tape around both the collector and emitter leads so that they won't come in contact and short out.

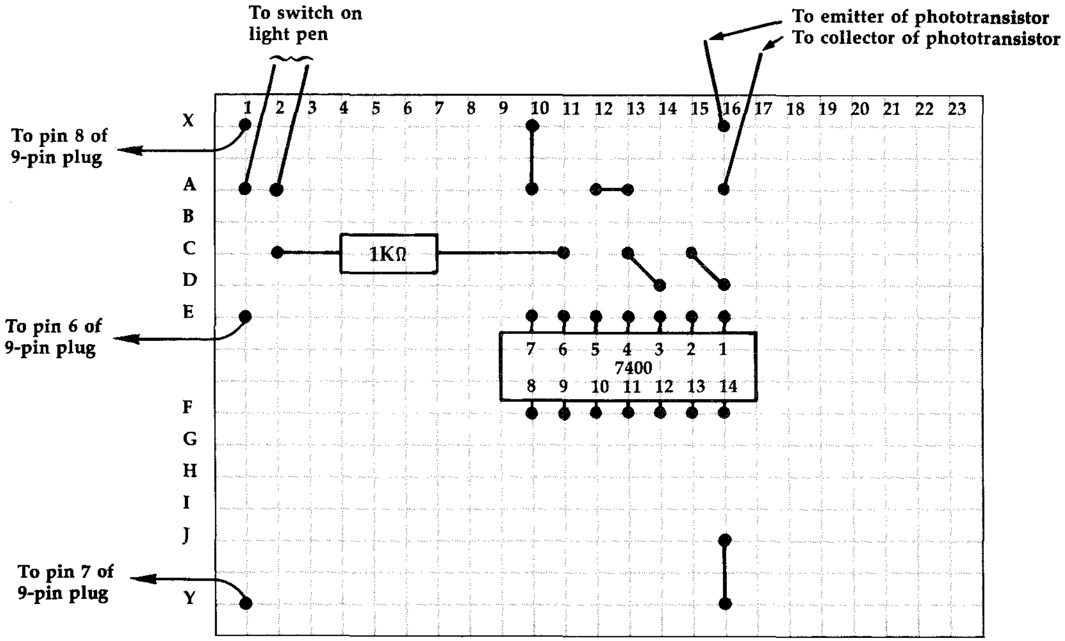
Figure 6-1. Light Pen Schematic Drawing


Q1
Pin 8
(GND)
U1A
7
1
3
2
14
U1B
4
6
5
R1
Pin 7+5v
S1
Pin 6

U_1	7400 quad 2-input NAND gate
Q_1	TIL 414 infrared phototransistor
R_1	1KΩ
S_1	Switch SPST (normally open)

This schematic of the light pen clearly shows how the various components are connected.

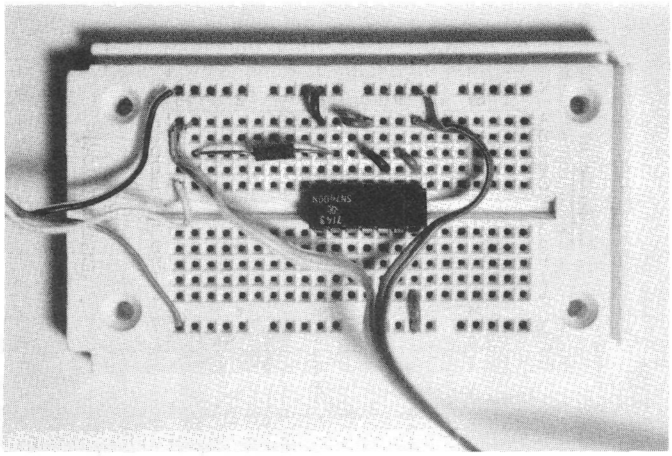
Figure 6-2. Solderless Breadboard Layout of Light Pen Circuit


To switch on light pen
To emitter of phototransistor
To collector of phototransistor
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
X
A
B
C
D
E
F
G
H
I
J
Y
To pin 8 of 9-pin plug
To pin 6 of 9-pin plug
To pin 7 of 9-pin plug
1KΩ
7 6 5 4 3 2 1
7400
8 9 10 11 12 13 14

Refer to this figure when wiring the breadboard.

8. Pull the wires of the phototransistor through the pen barrel and glue the phototransistor in place at the "tip" end. Be careful not to get any glue over the phototransistor's light-sensitive part.

Figure 6-3. In Reality—The Light Pen Breadboard



Schematics and layouts aside, this is how your light pen solderless breadboard should look when you're finished.

9. The wire from the emitter plugs into position X16 of the solderless breadboard, while the wire from the collector inserts into point A16.
10. Solder the remaining two 30-inch wires to the leads of the microswitch. Pull the wires through the hole you melted and out the open end of the pen's barrel. Glue the microswitch to the barrel of the pen.
11. One of the wires from the switch inserts into point A1 on the breadboard, while the other goes into point A2.
12. Insert the 1K resistor between points C2 and C11 of the solderless breadboard. (For the Atari version of this circuit, use a jumper wire in place of the resistor.)

To check that your light pen is operating properly, turn off your computer and insert the 9-pin plug into control port 1 of the Commodore 64/128 and Atari computers. (The VIC has but one control port.) A light pen won't work in control port 2 of the Commodore 64 and 128 computers. Turn on your computer, type in, and run the appropriate version of Program 6-1.

Program 6-1—Commodore 64/128

```
5  A=53267:B=53268
10 PRINT "X=";PEEK(A),"Y=";PEEK(B)
20 GOTO 10
```

Program 6-1—VIC-20

```
5  A=36870:B=36871
10 PRINT "X=";PEEK(A),"Y=";PEEK(B)
20 GOTO 10
```

Program 6-1—Atari

```
5  A=564:B=565
10 PRINT "X=";PEEK(A),"Y=";PEEK(B)
20 GOTO 10
```

Move your light pen around the screen while pressing the microswitch. The numbers of the two columns should change, giving you the coordinates of the point of your light pen.

Warning: You may have to adjust the brightness of your monitor to get your light pen to work properly.

How It Works

Your computer is set up to store the position of the electron beam every time pin 6 of the control port, normally at a logic high state, is set low. The light pen circuit is designed to set this pin low. This triggers the computer to save the electron beam's position when it passes the light pen.

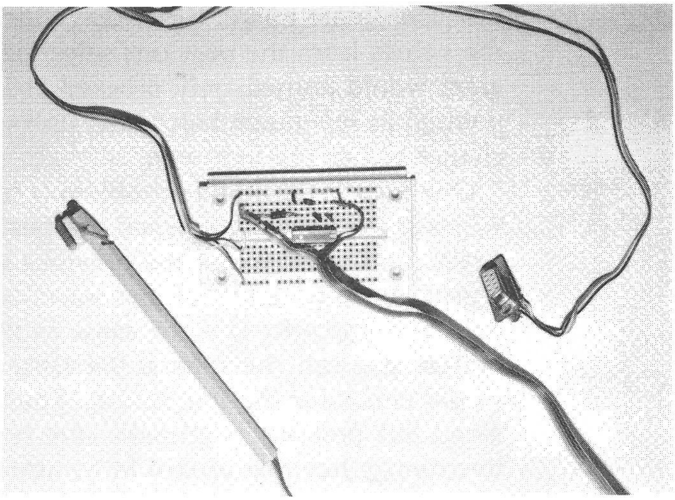
The phototransistor acts as a very fast switch. Normally, the path between the emitter and collector of the phototransistor is like that of an open switch. The moment the beam of the electron gun passes the point at which the phototransistor is aimed, enough light is emitted to turn the transistor on. When this happens, the connection between the emitter and the collector of this device is effectively that of a closed switch. Since the emitter is connected to ground, the inputs of the NAND gate, which are attached to the collector, are grounded as well.

The operation of NAND gates and other logic building blocks is explained in greater detail in Chapter 10, "Digital Logic." Let it suffice to say here that the two NAND gates used in the light pen circuit act as a *noninverting buffer.* Noninverting buffers are used when a signal isn't sufficient to drive the inputs of other IC gates. The output of this type of buffer is the same as its input.

The output of the buffer is connected to pin 6 of the control port through a resistor and the push button switch. When the push button switch is open, the light pen circuit is disconnected from pin 6 of the control port. The push button switch must thus be pressed if this pin is to be triggered.

When the light pen is aimed at a point on the screen and the button is pressed, several things happen. Because the screen picture is refreshed many times each second, the electron beam passes the phototransistor at least once in the time it takes to press and release the button. Before and after the beam passes the phototransistor, the input and output of the buffer are logic high, and as a result, pin 6 remains at that state. The moment the beam passes the phototransitor, the input and output of the NAND gate buffer are set low. Thus, pin 6 of the control port is set low for a fraction of a second, long enough to trigger the computer to store the location of the electron beam.

Figure 6-4. The Completed Light Pen



The entire light pen system, comprised of the breadboard, the ink-pen-based light pen, and the 9-pin connector, can be used for a multitude of programming applications, from drawing programs to menu selection.

Programming with the Light Pen

As you ran the test program, you probably noticed that when the pen was held at one spot, the coordinate numbers didn't remain constant, but instead stayed within a relatively small range. This must be taken into account when writing programs to work with your light pen.

Another point to remember is that once you use the light pen, the values remain the same in their registers until the light pen is used again. This means your program should have a way of checking that the values in the register are a *new* selection rather than a previous selection. For instance, suppose the registers contain the values 120 and

125 from a previous menu selection. When you put up your next menu, you expect the user to make another selection with the light pen. In order to determine where the light pen is pointed, you might compare the register values to a specified coordinate range. If the range includes one of the values from the previous selection, the program would immediately proceed as if the pen is pointed in *this* range before the user even has a chance to use the light pen.

One way to avoid this problem is to check whether the user has pressed the push button again. This insures that the previous values in the register have no effect on the selection. Since the light pen line (pin 6) is the same as that of the fire button, you can check for it the same way you would check for the fire button. Another way to avoid this problem is to make sure two consecutive menus have no ranges in common.

Try entering and running the appropriate version of Progam 6-2, which shows how a light pen can be used to make selections from a menu. When the menu appears, just point to the asterisk (*) beside your choice and press the microswitch. The screen clears and the choice you made appears. After a short delay, the word CONTINUE is displayed with an asterisk in front of it. Point your light pen to the asterisk and press the microswitch to return to the menu.

Program 6-2—Commodore 64/128

```
QK 100 REM * LIGHT PEN DEMO *
HA 150 A$=CHR$(147):PRINTA$
BB 160 PRINTTAB(10)"THREE CHOICES"
MQ 170 PRINT:PRINT
GJ 180 PRINTTAB(5)"* CHOICE ONE"
GF 190 PRINT:PRINT:PRINT:PRINT
DA 200 PRINTTAB(5)"* CHOICE TWO"
RC 210 PRINT:PRINT:PRINT:PRINT
FK 220 PRINTTAB(5)"* CHOICE THREE"
```

```
DD 230 A=PEEK(53267):B=PEEK(53268)
FJ 240 IF A>55 THEN 230
EB 250 IF (B>70) AND (B<80) THEN 300
HG 260 IF (B>110) AND (B<120) THEN 330
XP 270 IF (B>150) AND (B<160) THEN 360
HF 290 GOTO230
RF 300 PRINTA$
DA 310 PRINT"CHOICE ONE"
BF 320 GOTO 400
KH 330 PRINTA$
GD 340 PRINT"CHOICE TWO"
RJ 350 GOTO 400
FK 360 PRINTA$
QS 370 PRINT"CHOICE THREE"
KS 400 FOR I=0 TO 1500
XP 410 NEXT I:PRINT
CB 420 PRINT"* CONTINUE"
AS 430 IF PEEK(53267)<55 AND PEEK(53268)
       <70{2 SPACES}THEN 150
KX 440 GOTO 430
```

Program 6-2—VIC-20

```
XF 10 A$=CHR$(147):PRINTA$
JK 20 PRINTTAB(4)"THREE CHOICES"
CE 30 PRINT:PRINT
BQ 40 PRINTTAB(5)"* CHOICE ONE"
CF 50 PRINT:PRINT:PRINT:PRINT
HF 60 PRINTTAB(5)"* CHOICE TWO"
JC 70 PRINT:PRINT:PRINT:PRINT
FG 80 PRINTTAB(5)"* CHOICE THREE"
RH 90 A=PEEK(36870):B=PEEK(36871)
CA 100 IF A>65 OR A<40 THEN90
QQ 110 IF (B>30) AND (B<40) THEN150
DH 120 IF (B>50) AND (B<60) THEN180
HJ 130 IF (B>70) AND (B<80) THEN210
SX 140 GOTO90
MQ 150 PRINTA$
CK 160 PRINT"CHOICE ONE"
BR 170 GOTO230
GS 180 PRINTA$
HP 190 PRINT"CHOICE TWO"
AS 200 GOTO230
HX 210 PRINTA$
XE 220 PRINT"CHOICE THREE"
DF 230 FOR I=0 TO 1500
GS 240 NEXT I:PRINT
```

```
DP 250 PRINT"* CONTINUE"
QD 260 IF PEEK(36870)<40 AND PEEK(36871)
       <40{2 SPACES}THEN10
CF 270 GOTO260
```

Program 6-2—Atari

```
PF 100 REM LIGHT PEN DEMO
AN 150 PRINT CHR$(125)
ND 160 PRINT "THREE CHOICES"
LD 170 PRINT :PRINT :PRINT
CB 180 PRINT "* CHOICE ONE"
LF 190 PRINT :PRINT :PRINT
DC 200 PRINT "* CHOICE TWO"
KO 210 PRINT :PRINT :PRINT
LC 220 PRINT "* CHOICE THREE"
PH 230 A=PEEK(564):B=PEEK(565)
NB 240 IF A>90 THEN 230
CK 250 IF (B>25) AND (B<35) THEN 300
DC 260 IF (B>45) AND (B<55) THEN 330
DA 270 IF (B>60) AND (B<70) THEN 360
GJ 290 GOTO 230
AK 300 PRINT CHR$(125)
PC 310 PRINT "CHOICE ONE"
GC 320 GOTO 400
AN 330 PRINT CHR$(125)
AN 340 PRINT "CHOICE TWO"
GF 350 GOTO 400
BA 360 PRINT CHR$(125)
IO 370 PRINT "CHOICE THREE"
ME 400 FOR I=1 TO 15:PRINT :NEXT I
CC 410 PRINT
PG 420 PRINT "* CONTINUE"
DE 430 IF PEEK(564)<90 AND PEEK(565)
       >80 THEN 150
GI 440 GOTO 430
```

CHAPTER 7

A Digital Light Sensor

7

A Digital Light Sensor

The light pen you built in Chapter 6 is a type of light sensor—it detects light from the computer's screen and sends out an appropriate signal. But what if you want to be able to tell whether it's dark outside, or whether somebody opened your closet door (letting light inside)? Or perhaps you want to build an electronic timer that starts and stops timing when a beam of light is broken? To detect different levels of light, you need a slightly more complex sensor.

This chapter will show you how to build and use a *variable digital light sensor.* This sensor provides the computer with either a high (1) or low (0) digital signal. When the light sensor is adjusted for a certain level of light, the computer receives a logic low (0) signal as long as this level of light, *or more,* is maintained. However, if the level of light falls *below* this setting, the computer gets a logic high (1) signal from the sensor. This type of sensor is useful for applications such as light beam timers and counters, two applications which will be demonstrated here as well.

To build the digital light sensor, you need these parts:

Quantity	**Part**	**Part Number**
1	3900 op amp IC	276-1713
1	500K potentiometer (Commodore)	271-210
1	1M potentiometer (Atari)	271-211
1	100K ohm resistor	271-8045

Quantity	Part	Part Number
1	10K ohm resistor	217-8034
1	1K ohm resistor	271-8023
1	TIL 414 infrared phototransistor	276-130
1	9-pin D-subminiature female	276-1538
1	Solderless breadboard	276-175

You'll also need some solid copper wire for jumper connections and some stranded copper wire for connections to the phototransistor.

These steps will take you through the process of building your digital light sensor:

1. Cut three pieces of solid copper wire and remove about 1/4 inch of insulation from each end.
2. Connect one wire each to pins 1, 7, and 8 of the 9-pin plug.
3. Connect the wire from pin 8 to location X1 of the solderless breadboard.
4. Connect the wire from pin 7 to location Y1.
5. Connect the wire from pin 1 to location F5.
6. Plug the 3900 op amp IC into the solderless breadboard so that pin 1 goes to point E15, and pin 8 goes to point F9.
7. Cut two pieces of solid copper wire about 2 inches long and remove about 1/4 inch of insulation from each end. Solder one wire to an outside lead of the potentiometer. Solder the other wire to the middle lead of the potentiometer.
8. The wire from the outside lead connects to point X22 of the solderless breadboard. The wire from the potentiometer's middle lead connects to point A22.

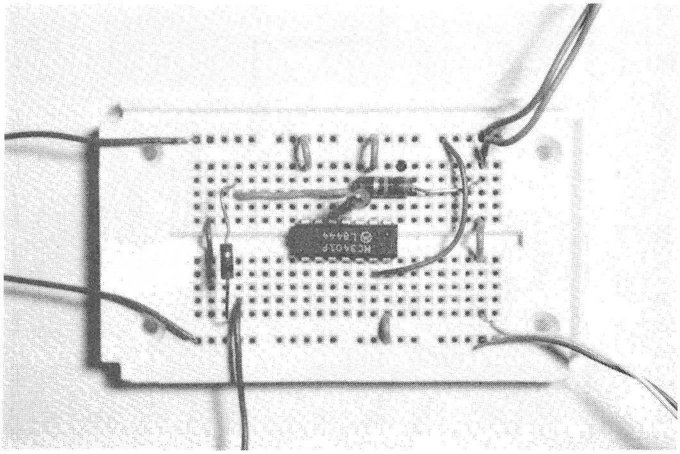
9. Cut two pieces of stranded copper wire about 12 inches long and remove about 1/4 inch of insulation from each end. Solder one wire to the emitter of the phototransistor and the other wire to the collector.
10. The wire from the collector plugs into point Y22 of the solderless breadboard, while the wire from the emitter inserts into point J22.
11. Insert the resistors as follows on the solderless breadboard:

Resistor	**From**	**To**
1K ohm	B4	J4
10K ohm	B22	B13
100K ohm	D13	D12

12. Connect five solid copper wire jumpers on the solderless breadboard as follows:

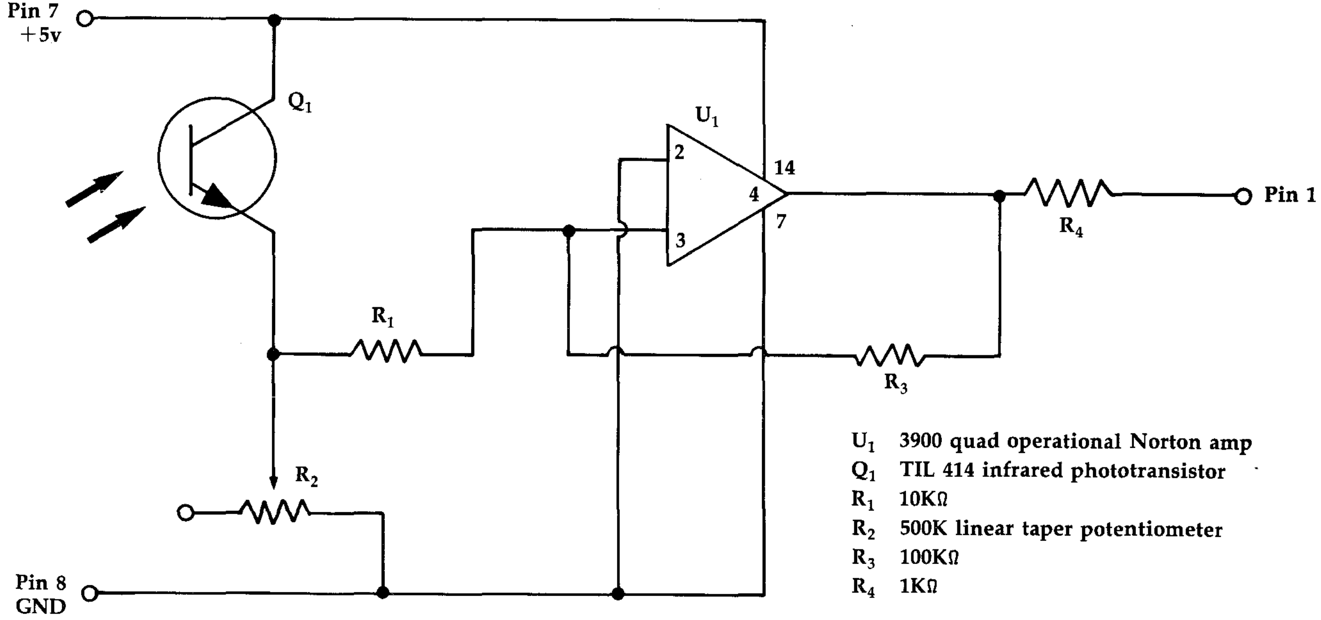
Jumper	**From**	**To**
1	E22	F22
2	Y15	J15
3	X14	A14
4	C12	C4
5	X9	A9

Figure 7-1. The Final Sensor Board



Your final light sensor breadboard will look like this.

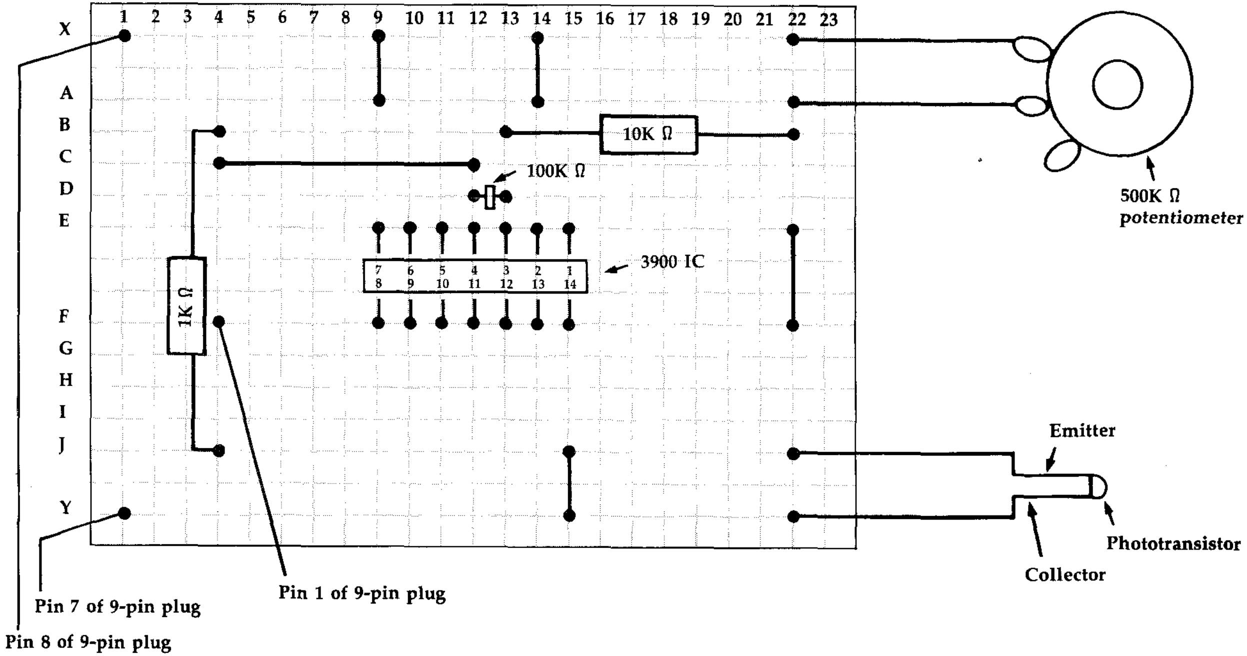
Figure 7-2. Schematic Drawing of Digital Light Sensor



U$_1$	3900 quad operational Norton amp
Q$_1$	TIL 414 infrared phototransistor
R$_1$	10KΩ
R$_2$	500K linear taper potentiometer
R$_3$	100KΩ
R$_4$	1KΩ

The digital light sensor is your most complex project yet. Notice the positions of the various resistors and the potentiometer. (Atari users—substitute a 1M potentiometer for the 500K potentiometer shown.)

Figure 7-3. Breadboard Layout for Light Sensor Circuit


1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
X
A
B
C
D
E
F
G
H
I
J
Y
10K Ω
100K Ω
3900 IC
7 6 5 4 3 2 1
8 9 10 11 12 13 14
1K Ω
500K Ω potentiometer
Emitter
Phototransistor
Collector
Pin 1 of 9-pin plug
Pin 7 of 9-pin plug
Pin 8 of 9-pin plug

All locations for wire and component placement are clearly marked. Atari users should substitute a 1M potentiometer for the 500K potentiometer shown.

Inside the Sensor

The phototransistor in this circuit acts as a switch. It's in series with the 500K potentiometer. When light strikes the light-sensitive surface, current will flow through the emitter-collector junction. The sensitivity is set by adjusting the 500K ohm potentiometer in series with the phototransistor. The operational amplifier's (op amp) inverting input is connected between the potentiometer and the phototransistor. The voltage at the op amp's input is determined by the resistances of the potentiometer and phototransistor.

The op amp is set up using the 10K ohm and 100K ohm resistors so that when the input voltage is a certain level, its output goes to the logic low level. The output of the op amp is connected to pin 1 of the control port through a 1K ohm resistor.

The phototransistor must be set for the desired level of light. This is done by aiming the light-sensitive part of the phototransistor at a light source, then adjusting the potentiometer for triggering at the desired light level. Program 7-1 provides the necessary lines for testing and adjusting this circuit.

Adjust the potentiometer so that the output to the computer turns logic low. Once this is done, if the phototransistor receives less light, the input voltage of the op amp no longer will be the correct voltage to force its output low. As a result, pin 1 of the 9-pin plug returns to its normal logic high level.

Putting It to Use

To use the light sensor, first turn off your computer. Insert the 9-pin plug into port 2 of the Commodore 64 or 128 (there's only one port on the VIC), or into port 1 of any Atari computer. Turn your computer back on, type in, and run the version of Program 7-1 for your machine. For this demonstration, the sensor is set to the lighting in the room, so make sure that the light-sensitive portion of the phototransistor is uncovered and facing up.

Program 7-1—Commodore 64/128

```
JE 10  IFPEEK(56320)AND(2↑0)THEN20
XR 15  PRINT"{CLR}LIGHT IS ON":GOTO10
EP 20  PRINT"{CLR}LIGHT IS OFF":GOTO10
```

Program 7-1—VIC-20

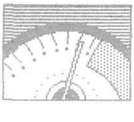
```
QD 10  IF(PEEK(37151) AND 4)THEN PRINT "
       {CLR}LIGHT IS OFF"
AS 20  IF(PEEK(37151) AND 4)=0THEN PRINT
       {SPACE}"{CLR}LIGHT IS ON"
SB 30  GOTO10
```

Program 7-1—Atari

```
10 IF STICK(0)=15 THEN PRINT “OFF”
20 IF STICK(0)=14 THEN PRINT “ON ”
30 GOTO 10
```

When you run the program, a column of messages should appear on the screen. Either the message LIGHT IS ON or LIGHT IS OFF will be displayed.

If the message is LIGHT IS ON, it means the sensor is detecting light at the level of or greater than the potentiometer setting. When the sensor detects the light level, it forces pin 1 of the control port low, just as connnecting it to pin 8 (ground) does. If the sensor does not detect the light level it's set for, pin 1 of the control port remains at logic high.



Adjust the setting of the potentiometer by turning its shaft. Doing this controls the sensitivity of the light sensor. You should find a point as you turn the potentiometer shaft where the messages in the column change from LIGHT IS OFF to LIGHT IS ON. In order to set the light sensor to detect the light in the room, leave the potentiometer set at the point where the messages just change from LIGHT IS OFF to LIGHT IS ON.

Just as with a joystick, programs can check the control port for the light sensor's signal. All that's necessary is to PEEK the appropriate register, or use the STICK command in the case of the Atari, then check the bit (corresponding to pin 1, in this case) for a 1 or 0.

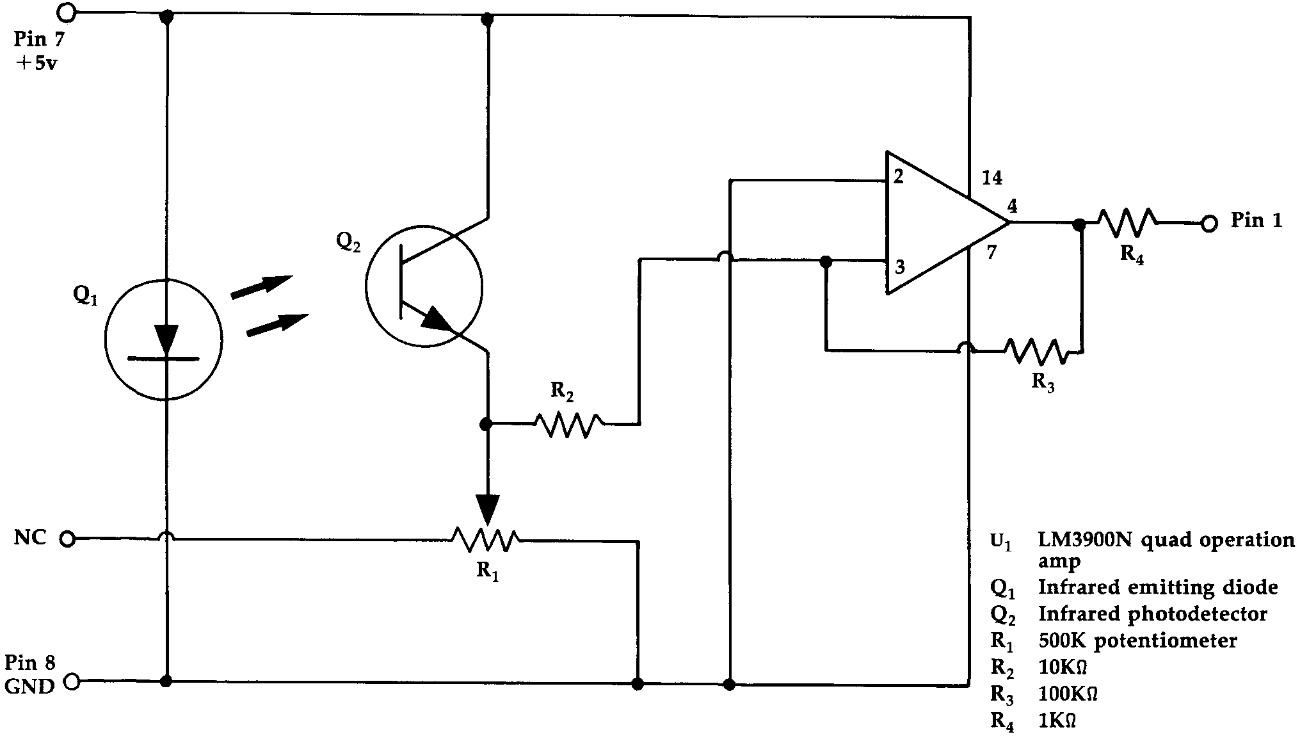
A Digital Light Beam Timer

This circuit, quite similar to the light sensor, can be used as a timer circuit for slot cars, or for any application involving measuring time over a distance. This project is essentially the same as the digital light sensor circuit you just completed. The following parts are required:

Quantity	**Part**	**Part Number**
1	3900 op amp IC	276-1713
1	500K potentiometer (Commodore)	271-210
1	1M potentiometer (Atari)	271-211
1	100K ohm resistor	271-8045
1	10K ohm resistor	217-8034
1	1K ohm resistor	271-8023
1	Infrared emitting diode	276-142
1	Infrared detecting diode	276-142
1	9-pin D-subminiature female	276-1538
1	Solderless breadboard	276-175

You'll also need some solid copper wire and some stranded copper wire.

Figure 7-4. Schematic of Digital Light Beam Timer Circuit


Pin 7
+5v
Q$_1$
Q$_2$
R$_2$
NC
R$_1$
Pin 8
GND
2
3
14
4
7
R$_4$
Pin 1
R$_3$

U$_1$	**LM3900N quad operational amp**
Q$_1$	**Infrared emitting diode**
Q$_2$	**Infrared photodetector**
R$_1$	**500K potentiometer**
R$_2$	**10KΩ**
R$_3$	**100KΩ**
R$_4$	**1KΩ**

This circuit is much like that of the digital light sensor except that the infrared emitting diode and the infrared detecting diode are used instead of a phototransistor.

Construct the circuit as you did for the digital light sensor, but position the infrared emitting diode and the infrared detecting diode so that the emitting surface and detecting surface face each other. Tests indicate that the maximum usable distance between these diode pairs (the two parts come in one package, listed as Radio Shack part number 276-142) is less than one foot.

Connect the infrared emitter's anode to the y line on the solderless breadboard, and the cathode to the x line. Use the appropriate version of Program 7-1 to adjust and focus the diodes so that the circuit triggers when the beam of light from the emitter is broken, *not* when ambient room light is interrupted. The light in the infrared is invisible to the naked eye. You'll see no indication that the diode is turned on. Be sure you've installed it correctly in the circuit.

When you've tested the circuit and are satisfied that it's functioning properly, type in and run Program 7-2. Adjust the potentiometer as you did with the digital light sensor, then press the X key to continue the program.

Start the timer by breaking the beam between the infrared emitting and infrared detecting diodes. When the beam is broken a second time, the elapsed time is displayed and the timer stops. Press the X key to reset the timer and run the program again.

This circuit could be useful as a burglar alarm, one that detects when an object or intruder breaks the beam of infrared light.

Program 7-2—Commodore 64/128

```
DX 110 REM TIMER PROGRAM
MH 120 PRINT"{CLR}ADJUST THE INFRARED BE
       AM AND THE"
FA 130 PRINT"POTENTIOMETER UNTIL THE MES
       SAGE JUST"
```

```
GQ 140 PRINT"CHANGES FROM OFF TO ON"
MG 150 PRINT:PRINT"PRESS X TO CONTINUE
       {DOWN}"
AQ 160 A$=CHR$(145)+CHR$(145)
PM 165 A=PEEK(56320)
JM 170 IF (A AND 1) THEN PRINT "LIGHT IS
        OFF"
GE 175 IF (A AND 1)=0 THEN PRINT "LIGHT
       {SPACE}IS ON "
KE 180 GET B$
BQ 190 IF B$="X"THENGOTO220
DS 200 PRINTA$
AD 210 GOTO 165
HD 220 PRINT"{CLR}BREAK BEAM TO START TI
       MER"
QX 230 IF (PEEK(56320) AND 1)=0 THEN 230
DF 240 TI$="000000":PRINT"{CLR}TIMER STA
       RTED"
MD 250 IF (PEEK(56320) AND 1) THEN 250
KH 260 IF (PEEK(56320) AND 1)=0 THEN 260
RC 270 PRINT"TIME IS";TI/60;"SECONDS"
XH 280 PRINT"{DOWN}PRESS X TO RESET TIME
       R"
DG 290 GETB$:IFB$=""THEN290
CD 300 IFB$="X"THEN220
MK 310 GOTO290
```

Program 7-2—VIC-20

```
SH 10 PRINT"{CLR}ADJUST THE INFRARED
      {3 SPACES}BEAM AND THE CIRCUIT
QS 20 PRINT"UNTIL THE MESSAGE JUST"
SR 30 PRINT"CHANGES FROM OFF TO ON"
KQ 40 PRINT:PRINT"PRESS X TO CONTINUE"
GM 50 A$=CHR$(145)
PB 60 A=PEEK(37137)
AE 70 IF (A AND 4) THEN PRINT "LIGHT IS
      {SPACE}OFF"
PQ 80 IF (A AND 4)=0 THEN PRINT "LIGHT I
      S ON "
XA 90 GET B$
MH 100 IF B$="X"THEN130
QB 110 PRINTA$A$
SQ 120 GOTO60
PH 130 PRINT"{CLR}BREAK BEAM TO START"
RP 140 IF (PEEK(37137)AND4)=0 THEN140
KS 150 TI$="000000"
RX 160 IF (PEEK(37137) AND 4) THEN160
JD 170 IF (PEEK(37137) AND 4)= 0 THEN170
```

```
MQ 180 PRINT"TIME IS";TI/60;"SECONDS"
HD 190 PRINT"PRESS X TO RESET TIMER"
XA 200 GETG$:IFG$=""THEN200
QX 210 GOTO130
```

Program 7-2—Atari

```
AP 110 REM TIMER PROGRAM
KP 120 PRINT "TURN ON FLASHLIGHT AND
        ADJUST THE"
LF 130 PRINT "POTENTIOMETER TILL THE
        MESSAGE JUST"
KO 140 PRINT "CHANGES FROM OFF TO ON
       "
FE 150 PRINT "PRESS X TO CONTINUE"
GH 155 FOR J=1 TO 2500:NEXT J
KE 170 IF STICK(0)=15 THEN PRINT "OF
       F"
GK 175 IF STICK(0)=14 THEN PRINT "ON
        "
MB 180 REM CHECK IF X KEY PRESSED
KI 190 IF PEEK(764)=22 THEN 215
GE 210 GOTO 170
BG 212 REM CLEAR OUT PREVIOUS KEYSTR
       OKE
DA 215 POKE 764,255
JB 220 PRINT "BREAK BEAM TO START TI
       MER"
IJ 230 IF STICK(0)=14 THEN 230
OA 240 POKE 18,0:POKE 19,0:POKE 20,0
IO 250 IF STICK(0)=15 THEN 250
IP 260 IF STICK(0)=14 THEN 260
BG 270 PRINT "TIME IS ";((PEEK(18)*2
       55*255)+(PEEK(19)*255)+(PEEK(
       20)))/60;" SECONDS"
```

For something a little different, modify Program 7-2 to count (COUNT = COUNT + 1) every time the beam is interrupted.

CHAPTER 8

An Electronic Switch

8 An Electronic Switch

So far, all the projects you've completed have been sensors—devices that transfer information about the outside world into your computer. But your computer can also be used to control outside devices through electronic signals.

While exploring the control port, you saw how information in the form of electronic on/off signals can be sent to your computer using the data (I/O) lines. These same wires can also be used by your computer to send digital signals *out.* Whether your computer is using the data lines for input or output, two registers are involved.

The first register, called the *data port,* contains the actual data being transferred. The signal on each data line corresponds with a bit stored at this location. When receiving data, your program simply checks the bit which corresponds to the affected I/O line. Suppose you wish to see whether pin 1 of control port 1 is set low, as it is if the joystick is pushed to the up position. For the Commodore 64/128, you could use the following statement:

```
10 IF (PEEK(56321) AND 2↑0)=0 THEN PRINT "YES"
```

On the VIC-20, you'd use this instead:

```
10 IF (PEEK(37137) AND 2↑0)=0 THEN PRINT "YES"
```

Since the zero bit in the memory location (the register) corresponds with pin 1 (the I/O line), the word YES is printed if this bit is set low.

When outputting data, your program must alter the contents of the data port and set the port for output. Suppose you want to set pin 1 of port 2 to output a logic high signal. To do this, your program must set bit 0 in the data direction register to a 1 for output:

For Commodore 64/128:

```
10 POKE 56321,PEEK(56321)OR(2↑0)
```

For Commodore VIC-20:

```
10 POKE 37137,PEEK(37137)OR(2↑0)
```

Each I/O pin of the control port has a corresponding bit in the data direction register. When a control port pin is to be used for input, its bit in this register must be set *low.* Setting a data direction register bit *high,* on the other hand, makes its corresponding control port pin an *ouput* line. The data lines are normally set for input—that's why you could ignore this register when you were first exploring the control port.

However, when you want the computer to output signals, this register must first be set. The data direction registers are located at memory locations 56322 ($DC02 in hexadecimal) for joystick port 2 and 56323 ($DC03) for joystick port 1 on the Commodore 64 and 128, and location 37139 ($9113) for the VIC-20.

The Atari Computers

The Atari computers are a little different when it comes to sending and receiving data. A data register and data direction register are still used, but unlike the Commodore computers, the Atari's registers are both at the same memory location.

Though you previously used the BASIC STICK command to check input to the control port, you can also use one of the computer's registers directly to obtain the same information. Bits 0–3 of memory location 54016 ($D300) correspond to pins 1–4 of control port 1, respectively, while bits 4–7 correspond to pins 1–4 of control port 2. By PEEKing this memory location, you can see what data has been input through the control ports. This same register is used to send information *out* of the computer using the control ports' lines. This is done by POKEing values into the register to set the bits corresponding to the control ports' pins either high or low. Memory location 54016 is used as a data register and contains the information being either sent or received.

Memory location 54016 acts as a data register, accepting incoming data, as long as bit 2 of memory location 54018 ($D302) is set to 0. If bit 2 of memory location 54018 is set to 1, location 54016 acts as a data direction register. Outgoing data can now be sent. The control ports' pins are set as either input or output lines, depending on whether their corresponding bits of the data direction register are 0's or 1's. You can set memory location 54016 to be the data direction register by

```
POKE 54018,48
```

and return it to acting as a data register with

```
POKE 54018,52
```

Note: With Atari BASIC, you cannot mask or force bits using ANDs and ORs in POKE statements as with Commodore computers. To set an individual bit in a memory location high, you can POKE the value directly, such as

POKE ***location*****, 2^*****bit***

where *location* is the memory address you wish to change and *bit* is the bit within that address that you wish to set. For example, to set bit 2 of location 54016, you would use

```
POKE 54016,4
```

Without AND and OR, there's no easy way to change a single bit without affecting the current settings of the other bits in the location. For example, if location 54016 already has bits 6 and 7 set (corresponding to a value of 192 in the location) and you wish to set bit 3 without affecting the others, then the value you must POKE into 54016 is 200, 192 + (2^3).

Using a Computer's Output Signals

How can an on/off signal from the computer control events in the outside world? An electronic circuit is required to take the digital signals that the computer outputs and perform some useful task. One example of this type of circuit is the electronic switch you'll make. It allows your computer to switch things on and off under the control of a computer program.

Building the Electronic Switch

You'll need these parts to construct your electronic switch:

Quantity	**Part**	**Part Number**
1	9-pin D-subminiature female	276-1538
1	2N2222 NPN transistor	276-1617
1	2.2K ohm resistor	271-8027
1	IN914 diode	276-1620
1	5-volt SPDT DIP relay	275-243
1	LED	276-041
1	Solderless breadboard	276-175
1	7400 IC (Atari)	276-1801

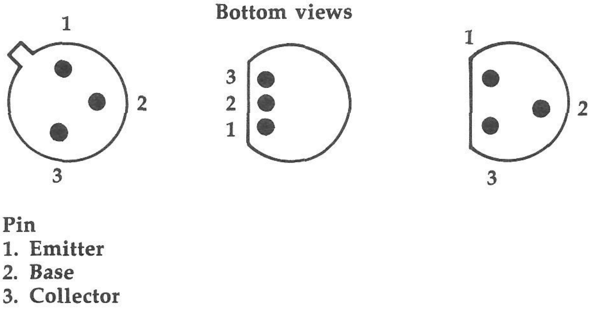
You'll also need some solid copper wire for connections to the 9-pin plug, as well as for jumper connections on the solderless breadboard.

Follow these steps to build your electronic switch if you have a Commodore computer (the steps are identical for the Atari, though some additional steps are listed below).

1. The first step in constructing the electronic switch is to wire the 9-pin plug. Cut three pieces of wire about four inches long and remove about 1/4 inch of the insulation from each end.

2. Solder wires to pins 1, 7, and 8.
3. Connect the three wires as follows: The wire from pin 8, the ground wire, connects to point Y1 on the solderless breadboard. The +5-volt wire from pin 7 connects to point X1 of the solderless breadboard. The data line from pin 1 connects to location B4 on the board (except for the Atari version—see below).
4. Connect the 2.2K ohm resistor between points A4 and F4 on the solderless breadboard.
5. Mount the 2N2222 transistor so that its base inserts into point H4, its emitter into point H3, and its collector into point H5.

Figure 8-1. Pin Diagrams for Typical 2N2222 NPN Transistors

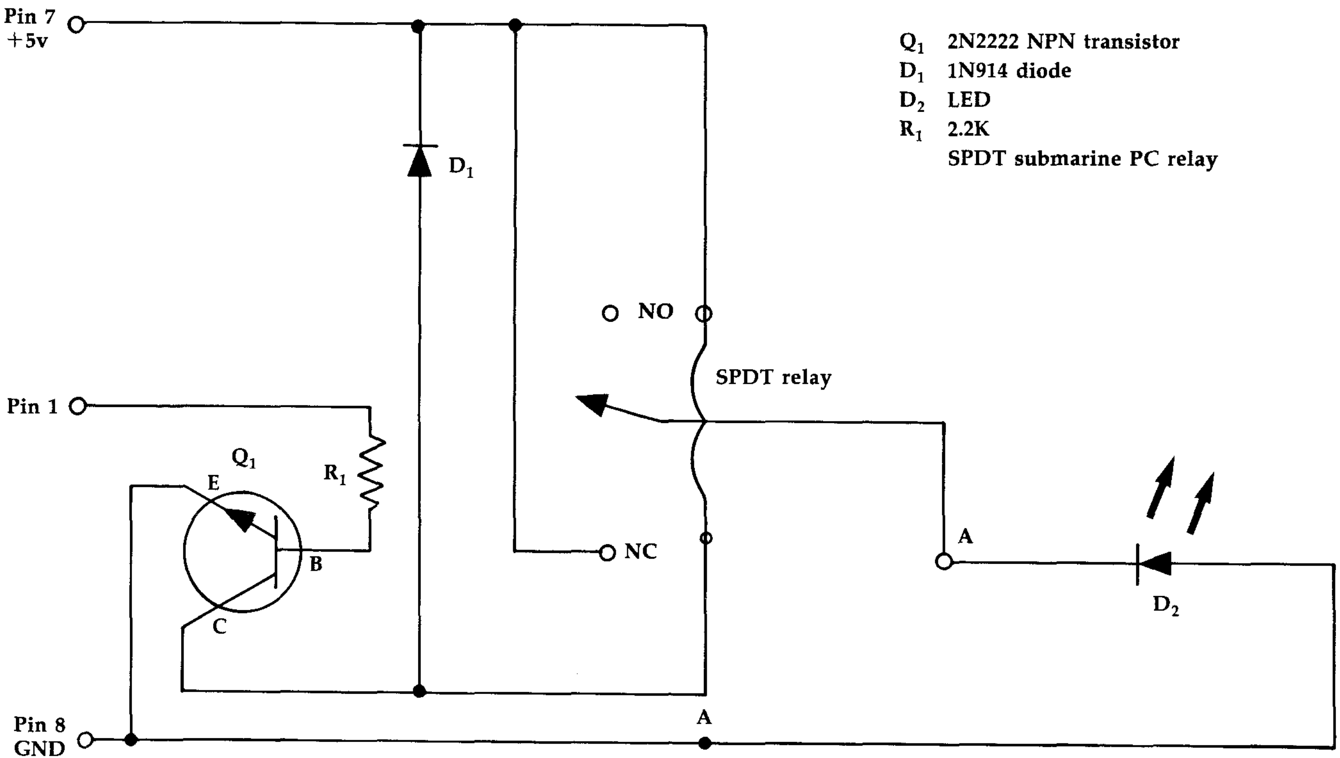


Here are three of the most common 2N2222 NPN transistor configurations you'll find.

6. The diode is connected so that its cathode lead inserts into point X5 of the solderless breadboard and its anode lead connects to point B5. (A band around one end of the diode is generally used to identify the cathode lead.)
7. The SPDT relay is inserted so that its normally open and normally closed pins attach to points F9 and E9 respectively. The common pin connects to point E14.

Continue with step 8 on page 88.

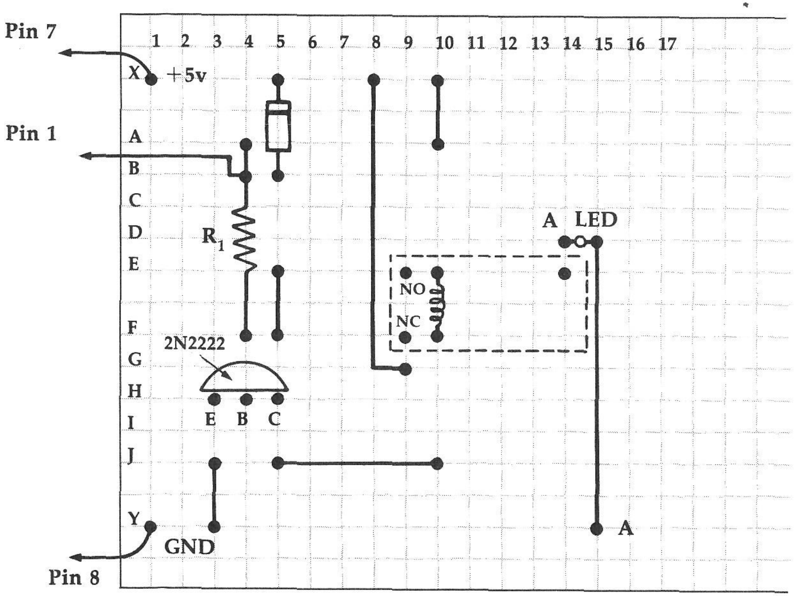
Figure 8-2. Schematic Drawing of the Electronic Switch


Pin 7
+5v
Pin 1
Pin 8
GND
D_1
Q_1
E
B
C
R_1
NO
NC
SPDT relay
A
A
D_2

Q_1	2N2222 NPN transistor
D_1	1N914 diode
D_2	LED
R_1	2.2K
	SPDT submarine PC relay

Schematic view of the electronic switch you'll be building.

Figure 8-3. Breadboard Layout of the Electronic Switch


Pin 7
Pin 1
Pin 8
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
X
+5v
A
B
C
D
E
F
G
H
I
J
Y
GND
R_1
2N2222
E B C
NO
NC
A LED
A

This is the breadboard layout for the Commodore computer version of the electronic switch.

8. The following jumper connections are required to complete the electronic switch.

Wire	From	To
1	J3	Y3
2	J5	J10
3	E5	F5
4	X10	A10

Additional Steps for Atari Version Only

9. Insert the 7400 IC into the solderless breadboard so that its pins 1 and 8 go into plug points F16 and E22 respectively.
10. Insert the wire from pin 1 of the 9-pin plug into point J16 of the solderless breadboard.
11. Add the following jumper connections.

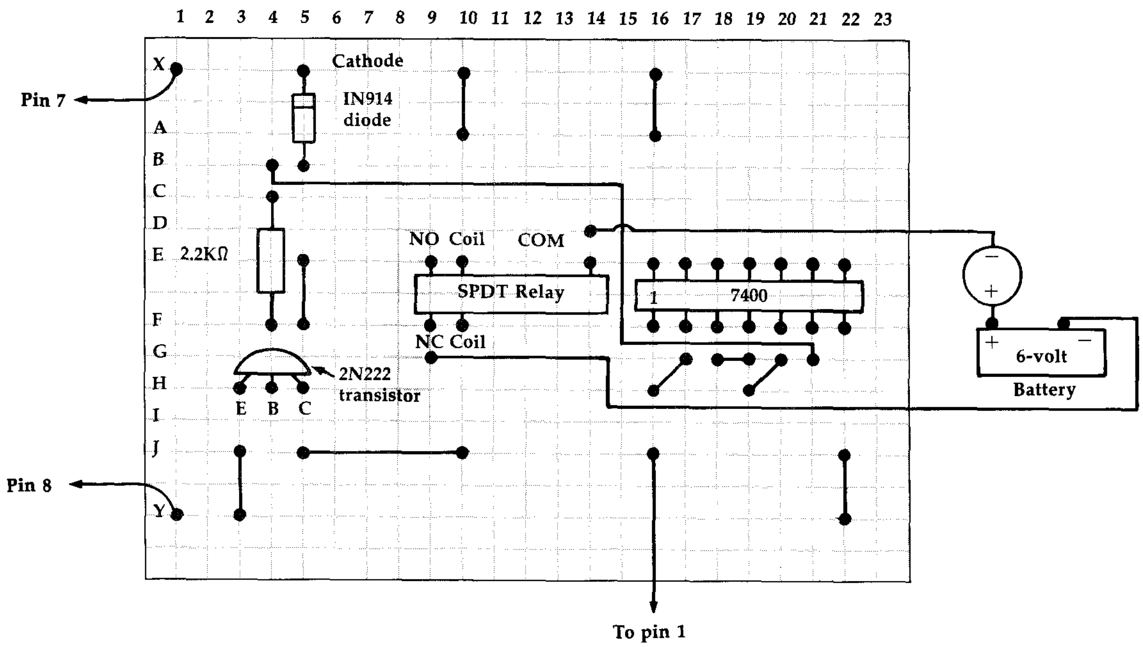
Wire	From	To
5	X16	A16
6	J22	Y22
7	H16	G17
8	G18	G19
9	H19	G20
10	G21	B4

Switch Is On

The 9-pin plug connects to control port 1 of your computer. The circuit draws its power from the computer's power supply through the wires from pins 7 and 8. The wire from pin 1 provides the circuit with a digital signal. This signal can be set by the program to be either logic high or logic low.

Pin 1 of the control port is connected to the base of the transistor through a current-limiting resistor. If the signal from the computer is +5 volts (logic high), current flows from this pin through the base of the transistor. This base current causes the transistor to act as a closed switch between its emitter and collector leads. When the

Figure 8-4. Breadboard Layout for Atari Electronic Switch


1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
X
A
B
C
D
E
F
G
H
I
J
Y
Pin 7
Pin 8
Cathode
IN914
diode
2.2KΩ
NO Coil
COM
SPDT Relay
NC Coil
1
7400
2N222
transistor
E B C
−
+
6-volt
Battery
To pin 1

Additional jumper wires and the 7400 IC must be installed on the breadboard for the Atari version of the switch.

transistor acts as a closed switch, current flows from the +5 volts of the power supply through the coil of the relay to ground. The relay turns *on* due to this coil current. The Atari version of the circuit uses two NAND gates as a noninverting buffer. This buffer is required since the signal from the Atari's control port is not sufficient to turn on the transistor. Refer to Chapter 10, "Digital Logic," for more on buffers and gates.

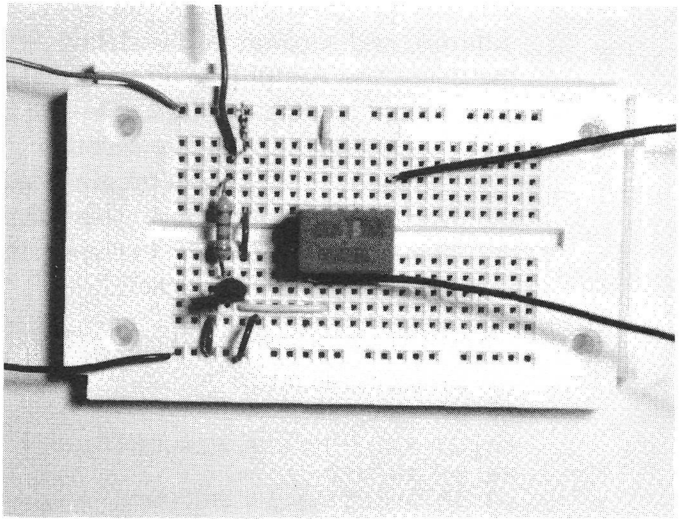
When a logic low signal (0 volts) is present at pin 1, no base current flows in the transistor. As a result, the transistor acts as an open switch across its emitter and collector leads. Consequently, the relay assumes its *off* position since no current flows through its coil.

The relay is an electromechanical switch. The device to be controlled is connected to the relay via its common, normally open and normally closed terminals. The circuit which controls the device is attached to the coil terminals of the relay.

When the relay is off, a metal lever bridges the gap between its common and normally closed terminals. The coil of the relay is part of an electromagnet. When current flows through this coil (to turn the relay on), the electromagnet pulls the metal lever so that it bridges the gap between the common and normally open terminals instead.

Many electric devices can be switched on and off by this circuit. A description of using the circuit to turn on an LED appears below. To use this circuit to control other devices, connect the devices between points G9 and D14, the two points marked *A* in Figure 8-3. Other devices should have their own power supply, such as a battery connected in series with point G9 or D14 and the device itself. (You can see the connection from the breadboard to a 6-volt battery in Figure 8-4.) Caution should be exercised not to exceed the current rating of the SPDT DIP relay.

Figure 8-5. Through with the Breadboard



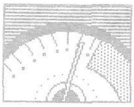
The electronic switch can be used to control a variety of devices.

Using the Electronic Switch—Flashing an LED

Here's one application for your electronic switch—a computer-controlled LED. The computer program, with the aid of the electronic switch, turns on the LED after a specified time delay. For this application, you'll need to connect two jumper wires and the LED.

1. Connect a wire between points X8 and G9 on your solderless breadboard.
2. Also connect a jumper between points Y15 and J15.
3. Plug the cathode of the LED into point D15 and its anode into D14.

First of all, insert the 9-pin plug of the electronic switch circuit into control port 1 of your computer. Next, turn on your computer.



Enter and run the appropriate version of Program 8-1. The program prompts you to enter the hours, minutes, and seconds of the delay. After you enter the delay, the computer will wait that length of time before turning on the LED—the LED will remain on for only about five seconds.

To turn on the LED, the program sets pin 1 of the control port to logic low. This turns off the relay, closing the connection between its common and normally closed terminals.

Program 8-1—Commodore 64/128

```
KR 10 PRINT"ENTER HOURS";
BK 20 INPUTA
SS 30 PRINT"ENTER MINUTES";
DP 40 INPUTB
SG 50 PRINT"ENTER SECONDS";
RR 60 INPUTC
GP 70 D=(A*216000)+(B*3600)+(C*60)
QE 80 REM START TIMER
QR 90 TI$="000000"
RM 100 IFTI>DTHEN130
QH 110 GOTO100
KX 120 REM SET DATA LINES FOR OUTPUT
JJ 130 POKE 56323,PEEK(56323)OR(2↑0)
DH 140 REM SET DATA LINES TO LOGIC LOW T
       O TURN ON BUZZER
CF 150 POKE 56321,PEEK(56321)ANDNOT(2↑0)
FX 160 REM KEEP BUZZER ON FOR FIVE SECON
       DS
AA 170 TI$="000000"
DK 180 IF TI>300 THEN210
HB 190 GOTO180
SQ 200 REM SET DATA LINE TO LOGIC HIGH T
       O TURN OFF BUZZER
XJ 210 POKE 56321,PEEK(56321)OR(2↑0)
FC 220 REM RESET DATA LINE FOR INPUT
QQ 230 POKE 56323,PEEK(56323)ANDNOT(2↑0)
```

Program 8-1—VIC-20

```
KR 10 PRINT"ENTER HOURS";
BK 20 INPUTA
SS 30 PRINT"ENTER MINUTES";
DP 40 INPUTB
```

```
SG 50  PRINT"ENTER SECONDS";
RR 60  INPUTC
GP 70  D=(A*216000)+(B*3600)+(C*60)
QE 80  REM START TIMER
QR 90  TI$="000000"
RM 100  IFTI>DTHEN130
QH 110  GOTO100
KX 120  REM SET DATA LINES FOR OUTPUT
BM 130  POKE 37139,PEEK(37139)OR(2↑2)
DH 140  REM SET DATA LINES TO LOGIC LOW T
        O TURN ON BUZZER
BD 150  POKE 37137,PEEK(37137)ANDNOT(2↑2)
FX 160  REM KEEP BUZZER ON FOR FIVE SECON
        DS
AA 170  TI$="000000"
DK 180  IF TI>300 THEN210
HB 190  GOTO180
SQ 200  REM SET DATA LINE TO LOGIC HIGH T
        O TURN OFF BUZZER
GM 210  POKE 37137,PEEK(37137)OR(2↑2)
FC 220  REM RESET DATA LINE FOR INPUT
RM 230  POKE 37139,PEEK(37139)ANDNOT(2↑2)
```

Program 8-1—Atari

```
IG 100 REM TIME BUZZER
KO 120 PRINT "ENTER HOURS";
GF 130 INPUT A
EE 140 PRINT "ENTER MINUTES";
GI 150 INPUT B
DA 160 PRINT "ENTER SECONDS";
GL 170 INPUT C
PP 180 D=(A*216000)+(B*3600)+(C*60)
OE 190 POKE 18,0:POKE 19,0:POKE 20,0
OJ 200 IF ((PEEK(18)*255*255)+(PEEK(
       19)*255)+(PEEK(20)))>D THEN 2
       20
FO 210 GOTO 200
NF 215 REM SET DATA LINES FOR OUTPUT
FN 220 POKE 54018,48
JA 225 POKE 54016,255
FJ 230 POKE 54018,52
MK 235 REM SET DATA LINES LOGIC LOW
       TO TURN ON BUZZER
CB 240 POKE 54016,0
OF 245 POKE 18,0:POKE 19,0:POKE 20,0
EC 250 IF ((PEEK(18)*255*255)+(PEEK(
       19)*255)+(PEEK(20)))>300 THEN
        270
```

```
GI 260 GOTO 250
ML 265 REM SET DATA LINES HIGH TO TU
       RN OFF BUZZER
JA 270 POKE 54016,255
AN 280 REM RESET DATA LINES FOR INPU
       T
GE 290 POKE 54018,48
BO 300 POKE 54016,0
FI 310 POKE 54018,52
```

Other Applications

The LED flasher is just one application of the electronic switch. You can use your electronic switch to turn on any electric device, anything from an alarm to a light, so long as the ratings of the relay and solderless breadboard are not exceeded. Simply wire the device and its power supply to the relay as if you were using an ordinary switch.

Warning: If you're planning on using the switch to turn on lights, a coffee pot, or any appliance connected to house line voltage, remember that the relay *must* be rated to handle the current needed by the appliance. *Potentially lethal voltages will be present at the relay.* The relay should be housed in a container to prevent accidental contact, and good electrical practices should be employed. For additional safety, a buffer circuit should be employed to further isolate your computer (and yourself) from high voltages.

CHAPTER 9

A Burglar Alarm

9 A Burglar Alarm

A burglar alarm is supposed to detect an intruder. To do that, though, the alarm must be as sophisticated as possible to detect and act upon a variety of situations. A computer, equipped with the proper sensors and actuators, can do just this and become an effective burglar alarm.

Intruders may reveal their presence to a computer-controlled burglar alarm in a number of ways. First of all, the intruder must somehow enter the building—a door or window is the most logical means of entry. If the computer's sensors are set to detect use of these entrances, an alarm can be triggered when someone passes through them.

Sensors could also be set to detect anyone in areas considered off-limits. For instance, if a sensor detects someone in the living room when all rightful occupants are sleeping upstairs, the burglar alarm can safely assume that an unauthorized person is in the protected area.

Another possibility is to have the alarm sensors detect movement of particular objects. Most car alarms work on this principle. The sensors can be set to tell whether your stereo, television, or anything else has been moved. If an intruder tries to steal something, the burglar alarm can alert you, or even alert the police.

Using What You Know

The previous projects in this book laid the groundwork for a burglar alarm. In particular, the joystick project in Chapter 3 and digital light sensor project in Chapter 7 will be used as the sensors. The electronic switch project, found in Chapter 8, is the actuator required by the alarm. If you didn't put them together earlier, turn back now and try them. Be sure you have a basic understanding of these circuits before proceeding. You'll need to refer to these chapters as you construct this burglar alarm.

As you saw in Chapters 2 and 3, a computer joystick is actually a set of switches. Many types of burglar alarm sensors are switches as well. These sensors can be connected to your computer in the same way as a joystick—through the I/O lines of the control port.

One type of switch sensor is a thin, adhesive, metallic tape that can be stuck on windows and other glass. This tape normally acts as a closed switch, allowing current to flow through a circuit. If the glass is broken, the fragile tape breaks, too, and the circuit is interrupted. Another type of switch sensor is a magnetic switch. This sensor is generally used to detect doors and windows being opened or closed, but it could also be installed on desk drawers. A typical application is a magnet mounted on a door, with the magnetic switch mounted on the door frame. When the door closes, the magnetic switch either closes or opens, depending on the type of switch. When the door opens, the magnetic switch changes.

Other types of sensor switches include vibration detectors, ultrasonic sensors, and even infrared sensors. Most of these "switch" alarm sensors can be purchased at electronic stores such as Radio Shack.

Your alarm will use the digital light sensor project built earlier to detect entry into an off-limits area. The light source and sensor should be set up so that any light in the controlled area will trigger the sensor.

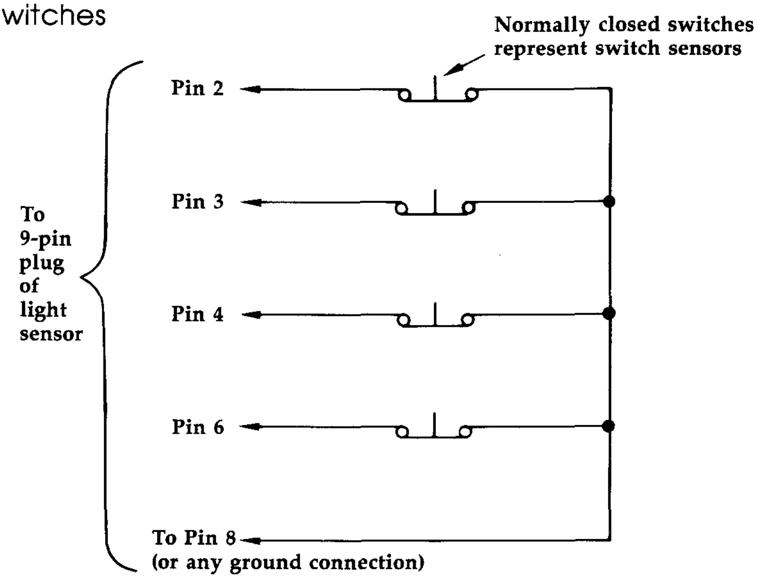
The computer will use the electronic switch to turn on the alarm itself. The alarm can be a buzzer or a louder alarm (providing it doesn't exceed the ratings of the relay or solderless breadboard). With the proper programming, your computer could even use a modem to call the police, and then play a tape recording announcing your address and reporting a break-in.

Setting Up the Alarm

To set up your computer as a burglar alarm, plug the electronic switch into control port 1.

Before placing the digital light sensor's 9-pin plug into control port 2, solder wires for switch sensors to its pins 2, 3, 4, and 6.

Figure 9-1. Atari and Commodore 64/128 Burglar Alarm Switches


Normally closed switches represent switch sensors
Pin 2
Pin 3
Pin 4
Pin 6
To 9-pin plug of light sensor
To Pin 8
(or any ground connection)

Notes
Digital light sensor plug inserts into control port 2.
Only additions to this circuit are sensors illustrated above.
Electronic switch circuit plugs into control port 1 unaltered.

The alarm sensors are wired to the digital light project's 9-pin plug.

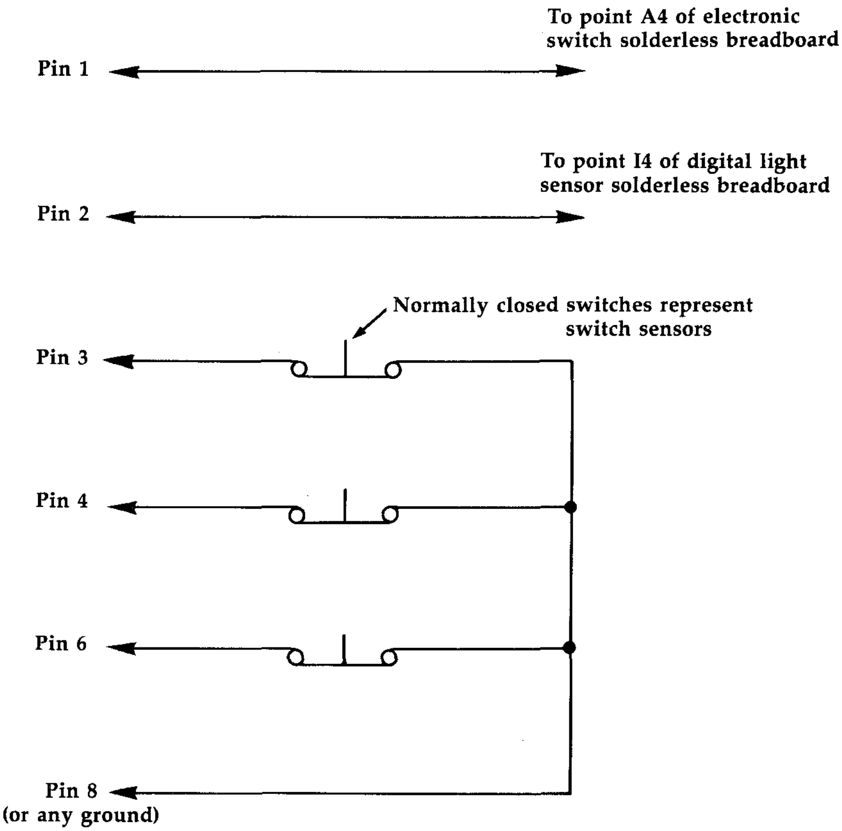
If you have a VIC-20, you need a new connector since both the electronic switch, the digital light sensor, and the other switch sensors must be plugged into the single control port.

To do this, you need to make a new connector:

1. Solder wires to pins 1, 2, 3, 4, and 6 of a 9-pin plug. Solder two wires each to pins 7 and 8.
2. Remove the old connectors from the electronic switch and light sensor cicuits.
3. The wire from pin 1 of the new connector inserts into point A4 of the electronic switch circuit. It provides the digital signal required to turn the electronic switch on or off.
4. The wire from pin 2 connects to point I4 of the light sensor circuit to receive its digital output signal.
5. The wires from pins 7 and 8 provide the power to the two circuits. One set of two wires from pins 7 and 8 connects to points Y1 and X1 respectively of the light sensor circuit board. The other set of two wires from pins 7 and 8 connects to points X1 and Y1 respectively on the electronic switch circuit. The remaining wires from the control port are used for switch sensors.

Whichever type (or types) of switch sensors you choose for your burglar alarm, they must be *normally closed*. They're set up so that they ordinarily form a closed loop between the ground pin of the control port and one of the I/O pins. The Commodore 64, 128, and Atari computers can each have four switch sensors connected to them in addition to the light sensor. The VIC-20 can have three switch sensors in addition to the light sensor. For any of the machines, there's no need to wire one end of the switch sensors directly to pin 8 of the control port—any connection to ground, such as the x row of the light sensor circuit, will do.

Figure 9-2. VIC-20 Burglar Alarm Switches


To point A4 of electronic switch solderless breadboard
Pin 1
To point I4 of digital light sensor solderless breadboard
Pin 2
Normally closed switches represent switch sensors
Pin 3
Pin 4
Pin 6
Pin 8 (or any ground)

Notes

One wire from pin 7 inserts to point Y1 of electronic switch solderless breadboard.

One wire from pin 8 inserts to point X1 of electronic switch solderless breadboard.

Second wire from pin 7 inserts to point X1 of digital light sensor.

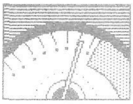
Second wire from pin 8 plugs into point Y1 of digital light sensor.

You must have special connector to build the burglar alarm for the VIC-20 since the computer has only one control port.

Since, in normal conditions, the switch sensors form a closed loop between ground and their particular I/O lines, these lines are at a logic low level. Similarly, the digital light sensor circuit outputs a logic high signal as long as no light is detected. The computer program of the burglar alarm can tell if things are as they should be just by checking to make sure that the bits in the data register corresponding to the sensors are off.

If one of the sensor bits of the data register has changed, this means one of the sensors has been activated—its corresponding I/O line no longer forced to ground. The burglar alarm program must detect if this happens and send a signal to the electronic switch to turn on the alarm.

You may want to experiment with infrared emitters and detectors, such as those used in the digital timer project of Chapter 7. In a darkened hallway, you might be able to use the digital timer as a switch for an intruder alert. Other infrared detectors and emitters could be purchased.



Program 9-1 is a simple demonstration program illustrating the use of your computer-controlled burglar alarm. Once you have the basic circuitry working, you can modify the program and circuit to add additional sensors as necessary. If you need more sensors than this simple alarm allows, refer to the multiplexer project of Chapter 12.

Program 9-1—Commodore 64/128

```
MH 120 PRINT"{CLR}ADJUST THE INFRARED BE
       AM AND THE"
FA 130 PRINT"POTENTIOMETER UNTIL THE MES
       SAGE JUST"
GQ 140 PRINT"CHANGES FROM OFF TO ON"
KX 150 PRINT:PRINT"PRESS X TO CONTINUE"
SF 160 A$=CHR$(145)
PM 165 A=PEEK(56320)
BJ 170 IF (A AND 2↑0) THEN PRINT "OFF"
XM 175 IF (A AND 2↑0)=0 THEN PRINT "ON "
```

```
KE 180 GET B$
FK 190 IF B$="X"THEN220
PH 200 PRINTA$A$
AD 210 GOTO165
EK 220 PRINT"TRIGGER A SENSOR TO START A
       LARM"
HQ 225 REM CHECK SENSOR STATUS
MX 230 B=PEEK(56320)
QH 240 IF B=A THEN 230
JQ 250 PRINT "WARNING-ALARM TRIGGERED"
AG 260 FOR I=0 TO 4
AA 270 IF (B AND 2↑I) THEN PRINT "SENSOR
        ";I+1;" DETECTS VIOLATION"
EH 280 NEXT I
MM 340 TI$="000000"
QM 370 REM TURN ON ALARM
SQ 375 REM SET DATA DIRECTION REGISTER F
       OR OUTPUT
XJ 380 POKE 56323,PEEK(56323)OR(2↑0)
AM 385 REM SET DATA LINE LOGIC LOW TO TU
       RN ON ALARM
HF 390 POKE 56321,PEEK(56321)ANDNOT(2↑0)
QC 395 REM ALARM ON FOR 2.5 SECONDS
KE 400 IF TI>3600/24 THEN 420
GP 410 GOTO 400
BP 420 REM SET DATA LINE LOGIC HIGH TO T
       URN OFF ALARM
AK 430 POKE 56321,PEEK(56321)OR(2↑0)
JR 440 REM RESET DATA DIRECTION REGISTER
        FOR INPUT
PM 450 POKE56323,PEEK(56323)ANDNOT(2↑0)
```

Program 9-1—VIC-20

```
GE 10 PRINT"TURN ON FLASHLIGHT AND ADJUS
      T THE"
GG 20 PRINT"POTENTIOMETER TILL THE MESSA
      GE JUST"
SR 30 PRINT"CHANGES FROM OFF TO ON"
KQ 40 PRINT:PRINT"PRESS X TO CONTINUE"
GM 50 A$=CHR$(145)
PB 60 A=PEEK(37137)
HS 70 IF (A AND 2↑3) THEN PRINT "OFF"
FH 80 IF (A AND 2↑3)=0 THEN PRINT "ON "
XA 90 GET B$
MH 100 IF B$="X"THEN130
QB 110 PRINTA$A$
SQ 120 GOTO60
```

```
DX 130 PRINT"TRIGGER A SENSOR TO START A
        LARM"
QG 140 Z$=" DETECTS VIOLATION"
MG 150 REM CHECK SENSOR STATUS
ES 160 B=PEEK(37137)
FS 170 POKE37154,0:IF(PEEK(37152)AND(2↑7
        ))THEN PRINT"SENSOR 3";Z$:GOTO230
EF 180 IF B=A THEN160
RP 190 PRINT "WARNING-ALARM TRIGGERED"
GG 200 IF (B AND 2↑3) THEN PRINT "SENSOR
         1";Z$
MM 210 IF (B AND 2↑4) THEN PRINT "SENSOR
         2";Z$
DX 220 IF (B AND 2↑5) THEN PRINT "SENSOR
         4";Z$
HB 230 POKE 37154,255
XE 240 TI$="000000"
PC 250 REM TURN ON ALARM
GH 260 REM SET DATA DIRECTION REGISTER F
        OR OUTPUT
HB 270 POKE 37139,PEEK(37139)OR(2↑2)
PR 280 REM SET DATA LINE LOGIC LOW TO TU
        RN ON ALARM
PM 290 POKE 37137,PEEK(37137)ANDNOT(2↑2)
RM 300 REM ALARM ON FOR 2.5 SECONDS
BC 310 IF TI>3600/24 THEN330
SG 320 GOTO310
HB 330 REM SET DATA LINE LOGIC HIGH TO T
        URN OFF ALARM
CB 340 POKE 37137,PEEK(37137)OR(2↑2)
AC 350 REM RESET DATA DIRECTION REGISTER
         FOR INPUT
EF 360 POKE37139,PEEK(37139)ANDNOT(2↑2)
```

Program 9-1—Atari

```
PL 110 REM ALARM PROGRAM
KP 120 PRINT "TURN ON FLASHLIGHT AND
         ADJUST THE"
LF 130 PRINT "POTENTIOMETER TILL THE
         MESSAGE JUST "
KO 140 PRINT "CHANGES FROM OFF TO ON
       "
FE 150 PRINT "PRESS X TO CONTINUE"
GH 155 FOR J=1 TO 2500:NEXT J
GP 160 IF STICK(1)=1 THEN PRINT "OFF
       "
DB 170 IF STICK(1)=0 THEN PRINT "ON
       "
```

```
MB 180 REM CHECK IF X KEY PRESSED
KD 190 IF PEEK(764)=22 THEN 210
GC 200 GOTO 160
ML 210 REM CLEAR OUT LAST KEYSTROKE
CM 220 POKE 764,255
DD 230 PRINT "TRIGGER A SENSOR TO ST
       ART ALARM"
NI 240 A=STICK(1):B=STRIG(1)
GM 250 IF (A=0 AND B=0) THEN 240
LG 260 PRINT "WARNING - ALARM TRIGGE
       RED"
OC 270 IF A=1 THEN PRINT "SENSOR 1"
OF 280 IF A=2 THEN PRINT "SENSOR 2"
OJ 290 IF A=4 THEN PRINT "SENSOR 3"
PA 292 IF A=8 THEN PRINT "SENSOR 4"
ON 294 IF B=1 THEN PRINT "SENSOR 5"
CF 300 PRINT "DETECTS VIOLATION"
NK 305 REM INITIALIZE CLOCK
NO 310 POKE 18,0:POKE 19,0:POKE 20,0
NC 320 REM SET DATA LINES FOR OUTPUT
FP 330 POKE 54018,48
IO 340 POKE 54016,255
FM 350 POKE 54018,52
OG 360 REM SET DATA LINES LOW TO TUR
       N ON ALARM
CF 370 POKE 54016,0
AM 380 IF ((PEEK(18)*255*255)+(PEEK(
       19)*255)+(PEEK(20)))>3600/24
       THEN 400
HA 390 GOTO 380
OF 400 REM TURN OFF ALARM BY SETTING
        DATA LINES LOGIC HIGH
IM 410 POKE 54016,255
AJ 420 REM RESET DATA LINES FOR INPU
       T
GA 430 POKE 54018,48
CD 440 POKE 54016,0
FN 450 POKE 54018,52
```

Before mounting sensors, you should first check your burglar alarm. The switch sensors can be simulated by connecting their I/O pin wires to the ground of either of the circuits. When you run the appropriate version of Program 9-1, you're instructed to carry out the necessary adjustment to the digital light sensor. Once you do this, your burglar alarm is operational.

To trigger the alarm, shine a light on the light sensor or disconnect a switch sensor wire from ground. The program prints on the screen which sensor was triggered and turns on the alarm after a delay of several seconds (so that an authorized person could disable the alarm). The alarm stays on for a couple of seconds before shutting off. You'll probably want to increase the delay of timers when you install the system. (Of course, a power failure will disable the alarm. The program will have been removed from your computer's memory.)

Other Applications

Not all the applications for this project are as serious as a burglar alarm. You could mount a magnetic switch on the refigerator door to make sure someone isn't cheating on a diet. If you have a small store, you could use this project to sound a buzzer when a customer enters. With some thought, you can modify the program and circuit for almost any application.

CHAPTER 10

Digital Logic

10 Digital Logic

Until now, whenever a digital output signal was required, you took it directly from one of the computer's control port I/O lines. Similarly, all the sensors sent their digital output signals directly to a control port pin. Every digital signal, whether input or output to the computer, needed its own I/O line.

Advanced projects cannot be interfaced to your computer like this—there simply are not enough I/O lines available. The control port gives you access to five I/O lines. But what if you need to control 12 electronic switches or monitor 15 sensors? Or even do both at the same time? The solution to this predicament is to use digital circuitry to manipulate the signals the computer sends and receives.

Digital circuitry is the same type of circuitry your computer uses. The uses of these types of circuits include performing calculations, storing information, and controlling the flow of digital signals. The behavior of digital circuits is described by what is known as *Boolean algebra.*

Even the most complex digital circuits, like the microprocessor of your computer, are created from basic building blocks. These building blocks are called *gates.* A gate has one or more input lines and only a single output line. The input lines are used to input binary data in the form of on/off signals. A gate combines these signals to produce an output. The output signal is either logic high (on) or logic low (off). The relationship between the input signals and the output signal of a gate is given by its *truth table.*

In the following descriptions, you'll see *H* and *L* listed. They represent:

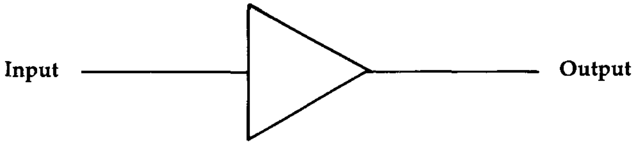
H = Logic high On 1 signal
L = Logic low Off 0 signal

The main types of gates, along with their truth tables, follow.

Noninverting Buffer

This is the simplest of all gates. The signal that appears at its input is the same as the signal on its output.

Figure 10-1. Symbol for a Noninverting Buffer

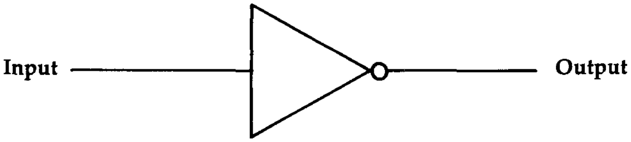


Input	**Output**
L	L
H	H

NOT Gate

NOT gates are often called *inverters*. A NOT gate's output signal is the opposite of its input signal. Note that the symbol of the NOT gate is the same as that of the buffer, except for the small circle near its output. Small circles at input or output lines of a gate's schematic symbol denote the fact that the signal is inverted.

Figure 10-2. Symbol for a NOT Gate

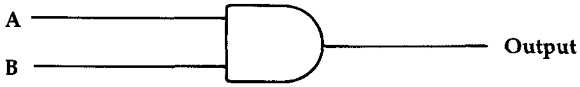


Input	Output
L	H
H	L

AND Gate

This gate has two or more inputs. Its output goes logic high *only* if all the inputs are logic high.

Figure 10-3. Symbol for an AND Gate

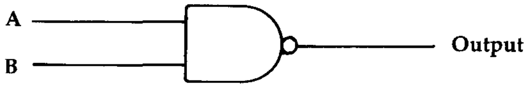


Input A	Input B	Output
L	L	L
L	H	L
H	L	L
H	H	H

NAND Gate

This gate has two or more inputs and only one output. Its output goes logic low only if all the inputs are logic high. For all other input conditions, the output is logic high. Note that the output of the NAND gate is the opposite of that of the AND gate for indentical input signals.

Figure 10-4. Symbol for a NAND Gate

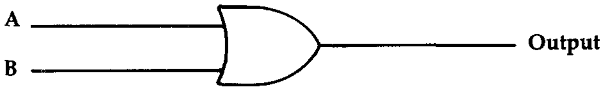


Input A	Input B	Output
L	L	H
L	H	H
H	L	H
H	H	L

OR Gate

The output of an OR gate goes high if any of its two or more input lines are set high.

Figure 10-5. Symbol for an OR Gate

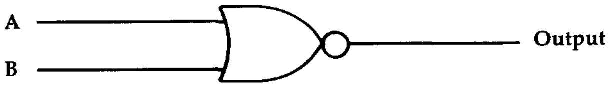


Input		**Output**
A	**B**	
L	L	L
L	H	H
H	L	H
H	H	H

NOR Gate

A NOR gate is identical to an OR gate with its output inverted. The output of a NOR gate is logic high only if *all* the inputs to the gate are logic low.

Figure 10-6. Symbol for a NOR Gate

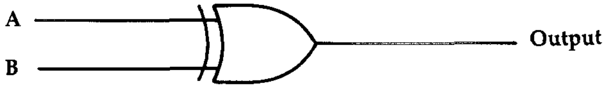


Input		**Output**
A	**B**	
L	L	H
L	H	L
H	L	L
H	H	L

Exclusive OR Gate

The output of this gate goes high if one input (but not more) goes logic high.

Figure 10-7. Symbol for an Exclusive OR Gate

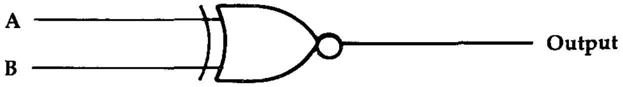


Input		Output
A	**B**	
L	L	L
L	H	H
H	L	H
H	H	L

Exclusive NOR Gate

The exclusive NOR gate has a logic low output when one (but not more) of the input lines is logic high. Its output is the same as that of an exclusive OR gate with its output inverted.

Figure 10-8. Symbol for an Exclusive NOR Gate

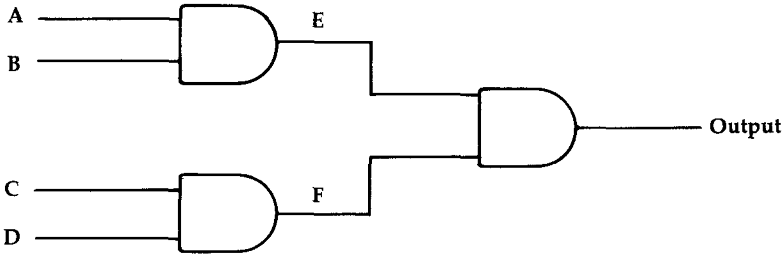


Input		Output
A	**B**	
L	L	H
L	H	L
H	L	L
H	H	H

Combining Gates

Logic gates can be combined in various ways. Two input gates can be combined to make three (or more) input gates. Figure 10-9 shows how three two-input AND gates can be used to produce a four-input AND gate.

Figure 10-9. Three Two-Input AND Gates Form a Four-Input AND Gate



Its function can be verified by using the following truth table:

Input				**Intermediate**		**Output**
A	**B**	**C**	**D**	**E**	**F**	
L	L	L	L	L	L	L
L	L	L	H	L	L	L
L	L	H	L	L	L	L
L	L	H	H	L	H	L
L	H	L	L	L	L	L
L	H	L	H	L	L	L
L	H	H	L	L	L	L
L	H	H	H	L	H	L
H	L	L	L	L	L	L
H	L	L	H	L	L	L
H	L	H	L	L	L	L
H	L	H	H	L	H	L
H	H	L	L	H	L	L
H	H	L	H	H	L	L
H	H	H	L	H	L	L
H	H	H	H	H	H	H

Logic gates can even be combined to form other logic gates. A NAND gate or a NOR gate acts as an inverter (NOT) if both the input lines are joined together.

Figure 10-10. NAND Gate As a NOT Gate

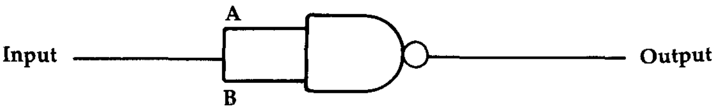
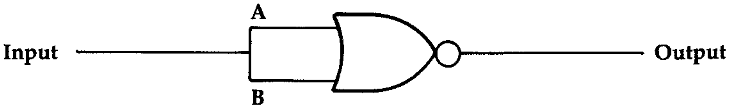
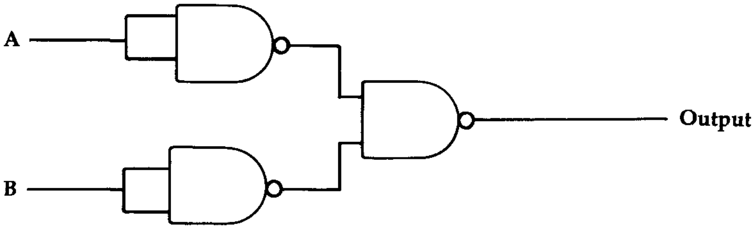


Figure 10-11. NOR Gate As a NOT Gate



Three two-input NAND gates can be hooked up to form an OR gate.

Figure 10-12. Three Two-Input NAND Gates As OR Gate



NAND and NOR gates are extremely useful since they can be wired to operate as any type of gate.

These gates can be combined in an unlimited number of ways. Once you understand the basic functions of the gates, putting together a digital circuit to meet specific requirements is fairly easy.

Decoders and Encoders

Also referred to as *demultiplexers* and *multiplexers, decoders* and *encoders* can be used by your computer to send or receive digital signals on many different lines using fewer pins of the control port. These circuits can help you solve the problem of controlling and sensing several different devices using a limited number of I/O lines. Like all digital circuits, decoders and encoders are constructed from the basic gates discussed earlier.

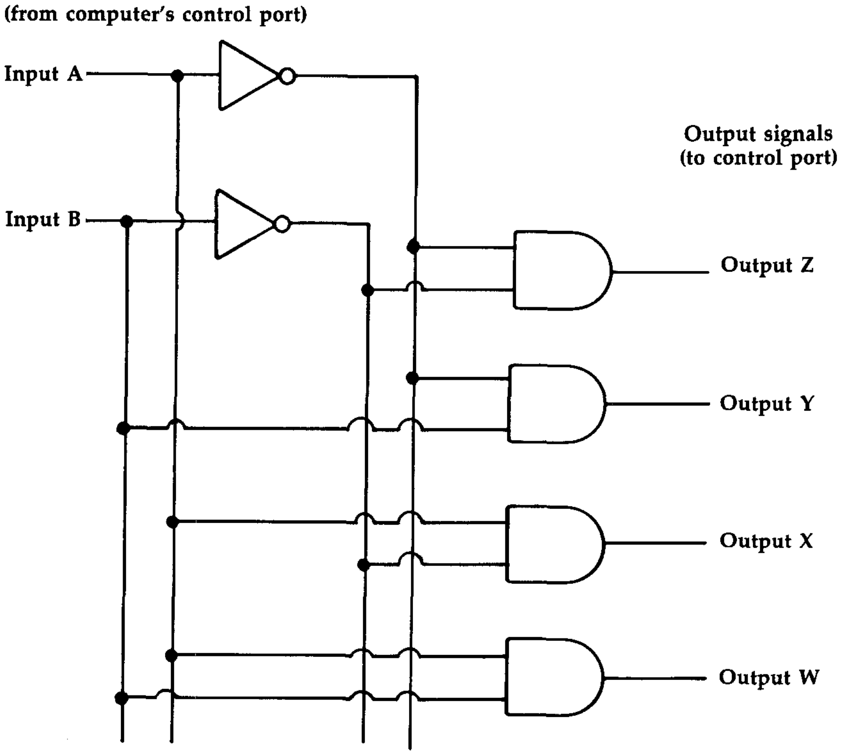
The decoder circuit in Figure 10-13 lets your computer output digital signals on four lines using only two lines of the control port. The two control port lines can output the binary numbers 00 (0 decimal) to 11 (3 decimal) in the form of high and low signals. One of the four output lines of the demultiplexer outputs a logic high signal depending on the input combination received from the control port. The other three outputs remain logic low. While the decoder allows you to turn on four electronic switches (or whatever) using two control port lines, there is a catch—you can switch on only one device at a time. In many applications this tradeoff between control port lines and flexibility will work to your advantage.

Try tracing the logic of the demultiplexer in Figure 10-13. You'll see that the inputs of the AND gates are wired to the input lines and inverted input lines from the control port in a way that always produces the expected output.

The number of outputs you can design in a decoder depends upon the number of control port lines you have available. If you have *X* control lines, you can make a demultiplexer with 2^X outputs. Of course, you would also need *X* inverters (NOT gates) and 2^X AND gates to build it.

Multiplexers, or encoders, perform the opposite function of decoders. They let you input digital signals from several lines to a single control port pin. As you can see from Figure 10-14, the circuitry for an encoder is similar to a decoder, with some additions. Each output of the *demultiplexer section* (the part on the left in Figure 10-14) is connected to an AND gate. The other lead of the AND gate is connected to one of the four lines carrying information from the sensors to the computer.

Figure 10-13. Decoder (Demultiplexer) Circuit


(from computer's control port)
Input A
Input B
Output signals
(to control port)
Output Z
Output Y
Output X
Output W

Truth Table

Input		Output			
A	**B**	**Z**	**Y**	**X**	**W**
L	L	H	L	L	L
L	H	L	H	L	L
H	L	L	L	H	L
H	H	L	L	L	H

The demultiplexer part of this circuit is used to select which of the four sensor lines is effectively connected to the control port pin. The line connected to the AND gate whose other input is logic high (from the demultiplexer part) has its signal passed to the OR gate and through to the control port. The signals from the lines connected to AND gates whose other inputs (from the demultiplexer section) are low are in essence blocked from the inputs of the OR gate.

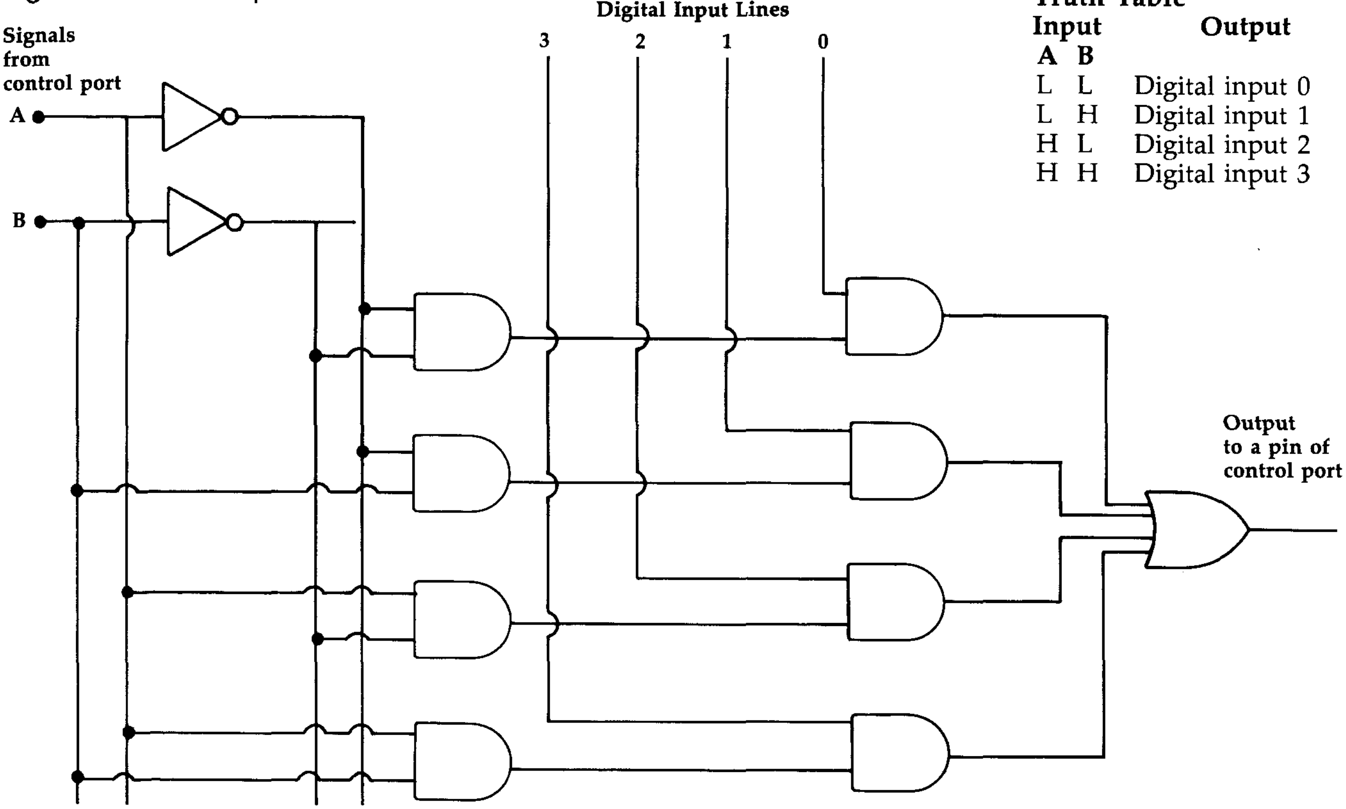
The only data line signal therefore "seen" by the control port is the one selected by the demultiplexer circuit. A multiplexer with X control port pins available can be designed to allow 2^(X−1) digital input lines to be connected to the computer. The value X−1 comes from the fact that one pin must be used to input the actual data, while the remaining pins are used to select the data line.

These encoder and decoder circuits may seem too large to set up on a solderless breadboard. Using individual chips, they would be. Fortunately entire multiplexers and demultiplexers are available on integrated circuit "chips." All you need to do is properly connect their pins to the control port. In fact, this procedure is described in Chapter 12, "More Ideas."

Experimenting with Digital Logic

The best way to learn and understand how digital circuitry works is to do some hands-on experimenting. All the different types of gates are available on integrated circuit (IC) chips. All you need are a few chips, a solderless breadboard, a power supply, some wire, and a logic probe to get started.

Figure 10-14. Multiplexer Circuit


Signals
from
control port
A
B
Digital Input Lines
3
2
1
0
Output
to a pin of
control port

Truth Table

Input		**Output**
A	**B**	
L	L	Digital input 0
L	H	Digital input 1
H	L	Digital input 2
H	H	Digital input 3

The integrated circuits you should use are TTL or LS types. These chips usually contain one or more gates. The 7400, for instance, is four two-input NAND gates in one IC package. It's a TTL (Transistor-Transistor Logic) IC. The 74LS00 is identical to the 7400 IC, except it uses less power (LS stands for Low-power Schotkey). In fact, most TTL ICs have LS versions.

TTL IC names start with the digits 74. Low-power Schotkey ICs follow this identifier with the letters *LS*. The remaining part of the names are the same for TTL and LS ICs which are functionally identical.

The pin layouts for the integrated circuits used in this book, as well as some additional basic ICs, appear in Appendix B.

Simply connect the ground and Vcc pins of the chip to a power supply. Be careful wiring the ICs to the power. If you reverse the connections, you could destroy the IC.

If you're using only a couple of chips, you can tap your computer's control port for power. Pin 7 can supply +5 volts with up to 50 mA (milliamperes) of current. Pin 8 is attached to the ground terminals of the computer's power supply.

If you're using several ICs, you'll need an external 5-volt power supply. When you use a separate external power supply for a digital circuit and interface it with your computer, remember to provide a common ground line by connecting the external power supply ground terminal to pin 8 of the control port.

CHAPTER 11

A Better Logic Probe

11 A Better Logic Probe

To experiment with digital logic circuits, you'll need a logic probe better than the simple one you built earlier. The logic probe illustrated here is capable of detecting both logic high and logic low states in a circuit. (If you're *really* serious about experimenting with digital circuits, you should definitely consider buying a commercial logic probe.)

To put together this more sophisticated logic probe, you'll need these components:

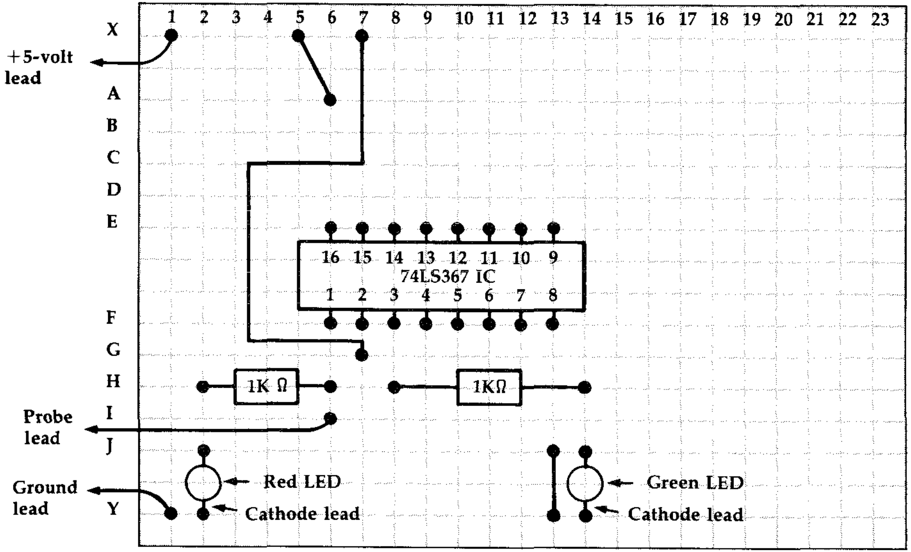
Quantity	**Part**	**Part Number**
2	1K ohm resistors	271-8023
1	Green LED	276-022
1	Red LED	276-041
3	Alligator clips	270-378
1	74LS367 IC	None
1	Solderless breadboard	276-175

You'll also need some solid copper wire for the jumpers and some stranded copper wire for probe leads.

Here's how to put together the "better" logic probe:

1. Cut three pieces of stranded copper wire about 8 inches long and remove about 1/4 inch of insulation from each end. Attach an alligator clip to one end of each of these wires.
2. Connect the other end of one wire to point Y1 of the solderless breadboard. This lead is the ground wire.

Figure 11-1. Solderless Breadboard Layout for Logic Probe


1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
X
A
B
C
D
E
F
G
H
I
J
Y
+5-volt lead
Probe lead
Ground lead
16 15 14 13 12 11 10 9
74LS367 IC
1 2 3 4 5 6 7 8
1K Ω
1KΩ
Red LED
Cathode lead
Green LED
Cathode lead

This more sophisticated logic probe uses two LEDs and a 74LS367 integrated circuit.

3. Insert one end of another wire, the +5-volt lead, to point X1 of the breadboard.
4. The third wire is the probe lead and is inserted into point I6 of the board.
5. Insert the 74LS367 IC into the breadboard so that its pin 1 goes into point F6, and its pin 9 goes into point E13.
6. Connect three jumper wires as follows:

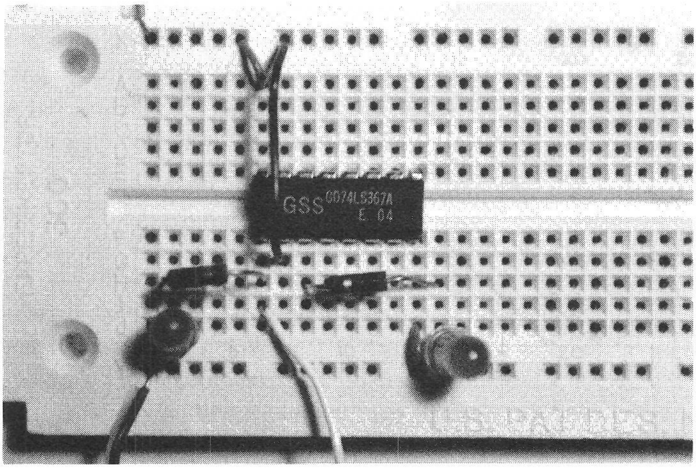
Jumper	From	To
1	G7	X7
2	J13	Y13
3	X5	A6

7. The 1K ohm resistors bridge the following points:

Resistor	From	To
1	H2	H6
2	H7	H14

8. The red LED is mounted on the breadboard so that its cathode plugs into point Y2 and its anode into point J2.
9. The cathode of the green LED inserts into point Y14, while its anode plugs into point J14.

Figure 11-2. Better Logic Probe



It's easy to see the various parts which make up this better logic probe.

Probing

To use the logic probe, connect the ground wire and +5-volt wire to the power supply ground and +5-volt terminals of the circuit you're checking. Touch the probe lead to the point of the circuit you want to check. If the green LED turns on, the point is at a logic low level, but if the red LED lights, the point is at a logic high level. (If neither LED lights, the point is neither at a logic high nor low state.)

When the probe is touched to a logic high point of the circuit, current flows through the red LED, causing it to light. The probe is also connected to the enable input pin of the 74LS367 IC. When this pin is high, all the buffer outputs in the chip enter the high impedance state. Since the green LED is connected to the output of a buffer, it remains off. The green LED is effectively disconnected from the circuit since one of its leads is connected to the buffer output, which is acting like a very large resistor (open circuit).

When the probe is touched to a logic low point of the circuit, no current flows through the red LED since both its leads are at the same voltage (ground). The low signal, however, enables the 74LS367 IC so that its buffers act normally. Since the buffer (whose output is connected to the green LED) input is attached to +5 volts, the green LED lights up. This happens because the output of the buffer is logic high, and as a result, a current flows through the green LED causing it to illuminate.

Note that due to low current levels, the LEDs may be dim and difficult to see at times.

This circuit is very handy for exploring and troubleshooting digital circuits. It's also a worthy candidate for "permanent" mounting on a printed circuit board, perhaps, as you'll probably use it often.

CHAPTER 12

More Ideas

12 More Ideas

You've built the simple circuits in this book, tried out the programs that accompanied them, and you're ready for more. Now what?

Since you've done the work already, let's take a look at some more ways to put these circuits to work. Once you start thinking about it, you'll quickly find that there's a very wide range of possible applications for these circuits. And the circuits can even be expanded to perform additional, more complex functions. Here are just a few projects you can create with the circuits you've built.

A Two-Beam Digital Timer Circuit

To make a two-beam digital timer, you'll need two digital light sensors (see Chapter 7).

The two-beam timer begins timing when the first light beam is broken and stops timing when the second light beam is broken. A light sensor is required to monitor each of these light beams. The elapsed time between the two events (breaking the paths of the two light beams) is displayed on the computer's screen. This project is ideal for timing races. It may also be useful in classroom science experiments where the speed of an object must be measured by timing how long it takes for something to move from point A to point B.

You've already built one sensor—the second sensor can be put on the same solderless breadboard as your original circuit. Both light sensors operate in the same way, and the digital sensors even share one major circuit component, the 3900 op amp IC. The 3900 IC has four operational amplifiers (op amps) built in. Each digital light sensor requires the use of one operational amplifier. The second sensor just uses one of the three unused op amps. In fact, you could build *four* digital light sensors using a single 3900 IC.

Figure 12-2 is the schematic diagram of the two-beam digital timer circuit. For a more reliable switch, infrared sources are recommended.

Assuming you have the first digital light sensor at hand (see Chapter 7), here's a list of additional parts you'll need to build this circuit.

Quantity	**Part**	**Part Number**
1	500K potentiometer (Commodore)	271-210
1	1M potentiometer (Atari)	271-211
1	100K ohm resistor	271-8045
1	10K ohm resistor	271-8034
1	1K ohm resistor	271-8023
1	TIL 414 infrared phototransistor	276-130

Here's how to construct your two-beam digital timer.

1. Take the solderless breadboard you used to create the digital light sensor in Chapter 7. Make sure all the components are still on the board and in their proper places.
2. Solder a 4-inch length of copper wire to pin 2 of the 9-pin plug. This wire will provide the computer with the digital signal from the second sensor.
3. Insert the other end of this wire into point J6 of the solderless breadboard.
4. Solder two wires to the middle and one of the outside leads of the 500K potentiometer.

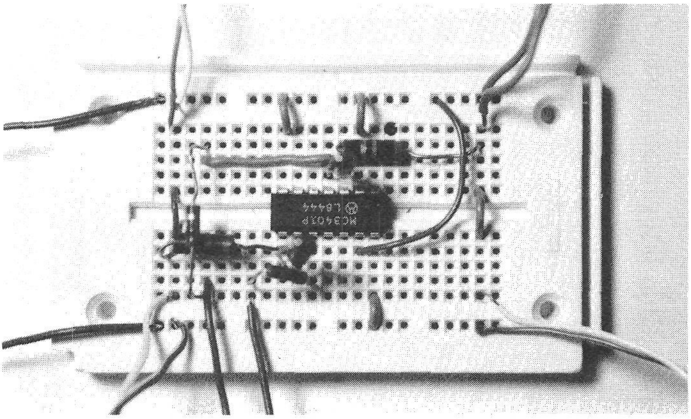
5. The other end of the middle wire goes to point A2 of the breadboard.
6. The wire from the outside lead goes to location X2 of the board.
7. Solder wires to the emitter and collector of the TIL 414 infrared phototransistor. The wires should be long enough so that the phototransistor can be mounted properly. Insert the emitter and collector leads into points J2 and Y2 respectively.
8. Bridge the three additional resistors between the following sets of points:

Resistor	**From**	**To**
R1 (10K ohm)	G2	G9
R3 (100K ohm)	H9	H10
R4 (1K ohm)	I6	I10

9. Make the following jumper wire connections (in addition to the jumpers which already exist on the digital light sensor solderless breadboard which you built in Chapter 7) to complete the hardware for the two-light-beam timer:

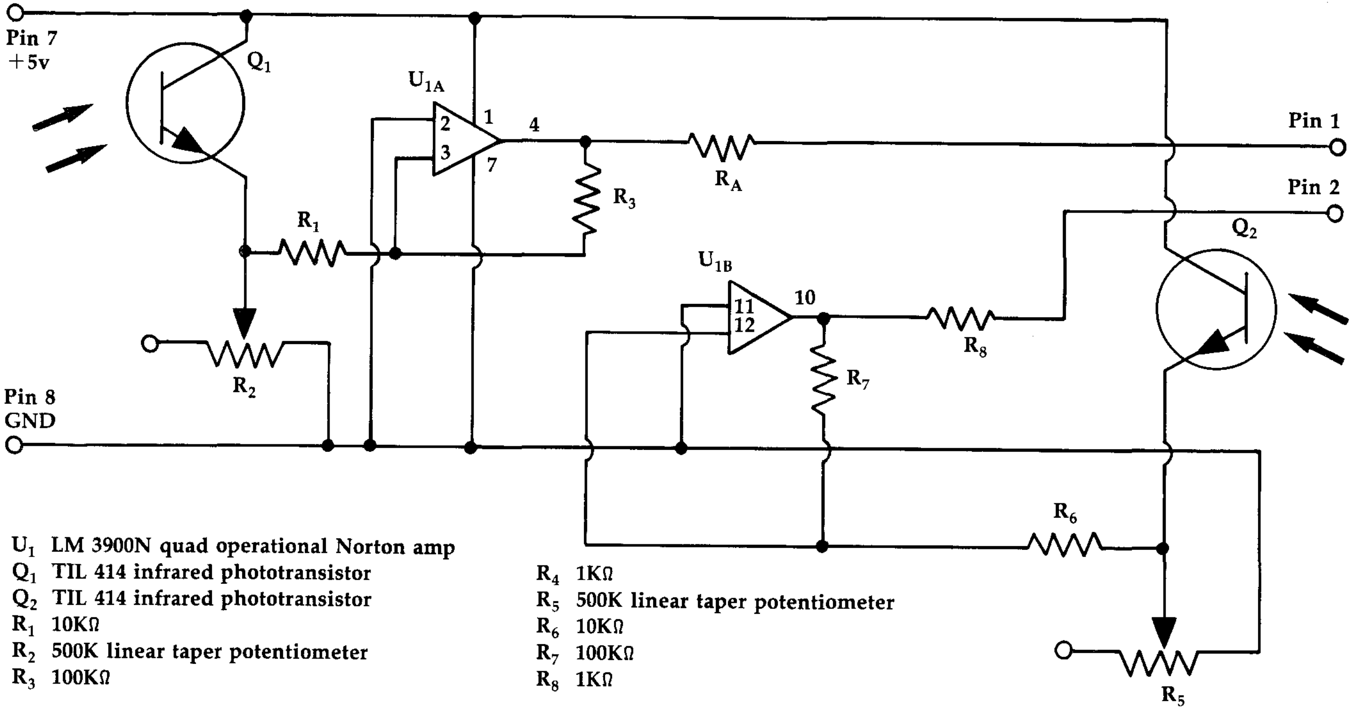
Jumper	**From**	**To**
1	G14	X19
2	F2	E2

Figure 12-1. Two-Beam Timer, the Final Look



This mass of wires and electronic components is actually a two-beam timer.

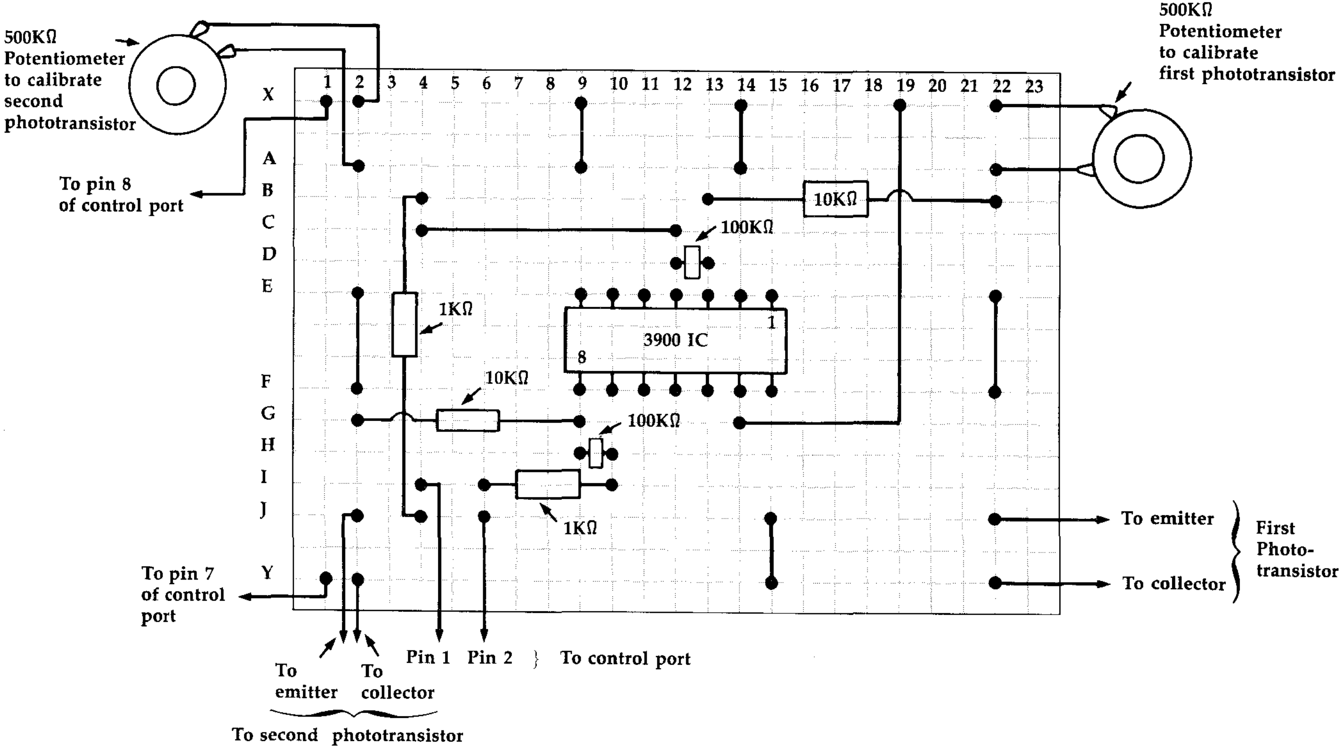
Figure 12-2. Schematic Drawing of the Two-Beam Digital Timer Circuit


Pin 7
+5v
Q$_1$
U$_{1A}$
2
3
1
4
7
R$_1$
R$_2$
R$_3$
R$_A$
Pin 8
GND
U$_{1B}$
11
12
10
R$_7$
R$_8$
Pin 1
Pin 2
Q$_2$
R$_6$
R$_5$

U$_1$	LM 3900N quad operational Norton amp
Q$_1$	TIL 414 infrared phototransistor
Q$_2$	TIL 414 infrared phototransistor
R$_1$	10KΩ
R$_2$	500K linear taper potentiometer
R$_3$	100KΩ
R$_4$	1KΩ
R$_5$	500K linear taper potentiometer
R$_6$	10KΩ
R$_7$	100KΩ
R$_8$	1KΩ

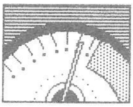
Though this schematic looks complicated, building the two-beam timer circuit is actually no more difficult than other projects you've already completed.

Figure 12-3. Two-Beam Timer Breadboard Layout


500KΩ
Potentiometer
to calibrate
second
phototransistor
To pin 8
of control port
To pin 7
of control
port
X
A
B
C
D
E
F
G
H
I
J
Y
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
10KΩ
100KΩ
1KΩ
3900 IC
1
8
10KΩ
100KΩ
1KΩ
500KΩ
Potentiometer
to calibrate
first phototransistor
To emitter
To collector
First
Photo-
transistor
To
emitter
To
collector
To second phototransistor
Pin 1
Pin 2
To control port

Using the breadboard circuit constructed in Chapter 7, you'll add several components and two jumper wires in building the two-beam digital timer.

Using the Two-Beam Timer



Plug the 9-pin connector into control port 2 for the Commodore 64/128, or into control port 1 for the Atari computers. To use the timer, set the sensors so that the light detect is interrupted when the object being timed passes them. Each sensor must be calibrated using its own potentiometer.

Program 12-1 monitors the two digital inputs to the computer from the sensors. When the first sensor is triggered by the passing object, it starts timing. When the object blocks the light to the second sensor, the program stops timing and displays the results.

Program 12-1—Commodore 64/128

```
JF 10 FOR I=0TO1
KF 20 PRINT"TURN ON FLASHLIGHT AND ADJUS
      T THE"
XP 30 PRINT"POTENTIOMETER ";I+1;" TILL T
      HE MESSAGE JUST"
DR 40 PRINT"CHANGES FROM OFF TO ON"
CR 50 PRINT:PRINT"PRESS X TO CONTINUE"
PM 60 A$=CHR$(145)
HM 70 A=PEEK(56320)
DX 80 IF (A AND 2↑I) THEN PRINT "OFF"
BE 90 IF (A AND 2↑I)=0 THEN PRINT "ON "
JS 100 GET B$
HE 110 IF B$="X"THEN140
JA 120 PRINTA$A$
CR 130 GOTO70
AS 140 NEXT I
BB 150 PRINT"BREAK BEAM 1 TO START TIMER
       "
GM 160 REM CHECK DATA REGISTER TILL FIRS
       T SENSOR TRIGGERED
MD 170 IF (PEEK(56320)AND(2↑0))=0 THEN17
       0
BH 180 REM START TIMER
JD 190 TI$="000000"
AG 200 REM CHECK DATA REGISTER TILL SECO
       ND SENSOR TRIGGERED
EM 210 IF (PEEK(56320) AND(2↑1))=0 THEN2
       10
KS 220 PRINT"TIME IS";TI/60;"SECONDS"
```

Program 12-1—VIC-20

```
SH 10 FOR I=2TO3
KF 20 PRINT"TURN ON FLASHLIGHT AND ADJUS
      T THE"
XP 30 PRINT"POTENTIOMETER ";I+1;" TILL T
      HE MESSAGE JUST"
DR 40 PRINT"CHANGES FROM OFF TO ON"
CR 50 PRINT:PRINT"PRESS X TO CONTINUE"
PM 60 A$=CHR$(145)
GB 70 A=PEEK(37137)
DX 80 IF (A AND 2↑I) THEN PRINT "OFF"
BE 90 IF (A AND 2↑I)=0 THEN PRINT "ON "
JS 100 GET B$
HE 110 IF B$="X"THEN140
JA 120 PRINTA$A$
CR 130 GOTO70
AS 140 NEXT I
BB 150 PRINT"BREAK BEAM 1 TO START TIMER
       "
GM 160 REM CHECK DATA REGISTER TILL FIRS
       T SENSOR TRIGGERED
BM 170 IF (PEEK(37137)AND(2↑2))=0 THEN17
       0
BH 180 REM START TIMER
JD 190 TI$="000000"
AG 200 REM CHECK DATA REGISTER TILL SECO
       ND SENSOR TRIGGERED
QC 210 IF (PEEK(37137) AND(2↑3))=0 THEN2
       10
KS 220 PRINT"TIME IS";TI/60;"SECONDS"
```

Program 12-1—Atari

```
FL 110 REM TWO LIGHT SENSOR TIMER PR
       OGRAM
FJ 112 REM CALIBRATE BOTH SENSORS
AI 115 FOR I=0 TO 1
KP 120 PRINT "TURN ON FLASHLIGHT AND
        ADJUST THE"
BE 130 PRINT "POTENTIOMETER ";I+1;"
       TILL THE MESSAGE JUST"
KO 140 PRINT "CHANGES FROM OFF TO ON
       "
BL 150 PRINT :PRINT "PRESS X TO CONT
       INUE"
GB 155 FOR J=1 TO 1000:NEXT J
BE 160 A=STICK(0)
KK 162 IF I=1 THEN 174
```

```
OJ 164 IF A=15 THEN PRINT "OFF"
OJ 166 IF A=13 THEN PRINT "OFF"
KO 168 IF A=14 THEN PRINT "ON "
KF 170 IF A=12 THEN PRINT "ON "
GN 172 GOTO 190
OK 174 IF A=15 THEN PRINT "OFF"
OL 176 IF A=14 THEN PRINT "OFF"
KO 178 IF A=13 THEN PRINT "ON "
KG 180 IF A=12 THEN PRINT "ON "
MC 190 REM CHECK IF X KEY PRESSED
JM 200 IF PEEK(764)=22 THEN 220
GD 210 GOTO 160
NE 220 REM CLEAR LAST KEYSTROKE
CN 230 POKE 764,255
BO 240 NEXT I
MF 250 PRINT "BREAK BEAM 1 TO START
       TIMER"
AO 260 REM CHECK FIRST SENSOR TILL T
       RIGGERED
MN 270 IF STICK(0)<>13 THEN 270
IN 280 REM START TIMER
OF 290 POKE 18,0:POKE 19,0:POKE 20,0
JE 300 REM CHECK SECOND SENSOR UNTIL
        TRIGGERED
II 310 IF STICK(0)=14 THEN 320
GG 315 GOTO 310
BC 320 PRINT "TIME IS ";((PEEK(18)*2
       55*255)+(PEEK(19)*255)+(PEEK(
       20)))/60;" SECONDS"
```

Light Switch

This project combines the analog light sensor from Chapter 5 with the electronic switch you built in Chapter 8. The analog light sensor detects the level of outside light. As night approaches, the computer detects the change in light conditions and turns on a light via the electronic switch. Near dawn, the sensor detects the increase in light, and the computer switches off the light. (Of course, other things, from dark clouds during the day to car headlights at night, may also activate the switch.)

The analog light sensor, a cadmium sulfide photocell, is connected between pins 9 and 5 of the electronic switch's 9-pin plug. Use leads long enough so that the sensor can be aimed easily. The electric light is connected to the relay of the electronic switch across its normally closed and common terminals (remember, the relay turns *on* as soon as you plug the circuit into the computer's port). Be sure that the ratings of the solderless breadboard or relay are not exceeded by the electric light. The circuit is connected to control port 1 of your computer.

Program 12-2—Commodore 64/128

```
GX 10 PRINT "SET LIGHTING LEVEL TO SWITC
      H ON"
EG 20 PRINT "PRESS X TO CONTINUE"
RA 30 A=PEEK(36872)
HR 40 GET B$
RR 50 IF B$="X" THEN70
XE 60 GOTO30
QG 70 REM
HP 80 PRINT "SET LIGHTING LEVEL TO SWITC
      H OFF"
PD 90 PRINT "PRESS X TO CONTINUE"
AX 100 B=PEEK(36872)
ES 110 GET B$
DF 120 IF B$="X" THEN140
EJ 130 GOTO100
MJ 140 PRINT "PRESS X TO START"
FB 150 GET B$
HQ 160 IF B$="X" THEN180
PS 170 GOTO150
PR 180 REM CHECK IF LIGHT LEVEL LOWER TH
       AN FIRST SETTING
FK 190 IF PEEK(36872)<A THEN190
SP 200 REM SET DATA DIRECTION REGISTER F
       OR OUTPUT
XG 210 POKE 37139,PEEK(37139)OR(2↑2)
CC 220 REM SET OUTPUT LINE LOGIC LOW
CG 230 POKE 37137,PEEK(37137)ANDNOT(2↑2)
HQ 240 REM CHECK IF LIGHTING LEVEL HIGHE
       R THAN SECOND SETTING
FP 250 IF PEEK(36872)>B THEN250
HQ 260 REM SET OUTPUT LINE LOGIC HIGH
```

```
RG 270 POKE 37137,PEEK(37137)OR(2↑2)
DG 280 REM RESET DATA DIRECTION REGISTER
        FOR INPUT
FB 290 POKE 37139,PEEK(37139)ANDNOT(2↑2)
```

Program 12-2—VIC

```
GX 10 PRINT "SET LIGHTING LEVEL TO SWITC
      H ON"
EG 20 PRINT "PRESS X TO CONTINUE"
HD 30 A=PEEK(54297)
HR 40 GET B$
RR 50 IF B$="X" THEN70
XE 60 GOTO30
QG 70 REM
HP 80 PRINT "SET LIGHTING LEVEL TO SWITC
      H OFF"
PD 90 PRINT "PRESS X TO CONTINUE"
MB 100 B=PEEK(54297)
ES 110 GET B$
DF 120 IF B$="X" THEN140
EJ 130 GOTO100
MJ 140 PRINT "PRESS X TO START"
FB 150 GET B$
HQ 160 IF B$="X" THEN180
PS 170 GOTO150
PR 180 REM CHECK IF LIGHT LEVEL LOWER TH
       AN FIRST SETTING
CM 190 IF PEEK(54297)<A THEN190
SP 200 REM SET DATA DIRECTION REGISTER F
       OR OUTPUT
ED 210 POKE 56323,PEEK(56323)OR(2↑0)
CC 220 REM SET OUTPUT LINE LOGIC LOW
CJ 230 POKE 56321,PEEK(56321)ANDNOT(2↑0)
HQ 240 REM CHECK IF LIGHTING LEVEL HIGHE
       R THAN SECOND SETTING
BE 250 IF PEEK(54297)>B THEN250
HQ 260 REM SET OUTPUT LINE LOGIC HIGH
GD 270 POKE 56321,PEEK(56321)OR(2↑0)
DG 280 REM RESET DATA DIRECTION REGISTER
        FOR INPUT
ED 290 POKE 56323,PEEK(56323)ANDNOT(2↑0)
```

Program 12-2—Atari

```
LP 100 REM LIGHT SWITCH
HG 110 REM
FC 120 PRINT "SET LIGHT LEVEL TO SWI
       TCH ON"
FC 130 PRINT "PRESS X TO CONTINUE"
DO 140 A=PADDLE(0)
LO 150 REM CHECK IF X KEY PRESSED
KG 160 IF PEEK(764)=22 THEN 180
GG 170 GOTO 140
NN 175 REM CLEAR LAST KEYSTROKE
DB 180 POKE 764,255
HF 190 PRINT "SET LIGHTING LEVEL TO
       SWITCH OFF"
FA 200 PRINT "PRESS X TO CONTINUE"
DN 210 B=PADDLE(0)
LM 220 REM CHECK IF X KEY PRESSED
KD 230 IF PEEK(764)=22 THEN 260
GC 240 GOTO 210
NH 250 REM CLEAR LAST KEYSTROKE
DA 260 POKE 764,255
EI 262 PRINT :PRINT "PRESS X TO STAR
       T"
LC 264 IF PEEK(764)=22 THEN 268
HD 266 GOTO 264
DI 268 POKE 764,255
LE 270 REM CHECK IF LIGHT LEVEL LOWE
       R THAN FIRST SETTING
JK 280 IF PADDLE(0)<A THEN 280
GD 290 REM SET DATA DIRECTION REGIST
       ER FOR OUTPUT
FM 300 POKE 54018,48
IL 310 POKE 54016,255
FJ 320 POKE 54018,52
FL 330 REM SET DATA LINES LOGIC LOW
CC 340 POKE 54016,0
PD 350 REM CHECK IF LIGHTING LEVEL H
       IGHER THAN SECOND SETTING
JN 370 IF PADDLE(0)>B THEN 370
IO 380 REM SET DATA LINES LOGIC HIGH
JD 390 POKE 54016,255
AH 400 REM RESET DATA LINES FOR INPU
       T
FO 410 POKE 54018,48
CB 420 POKE 54016,0
FL 430 POKE 54018,52
```

Enter and run the appropriate version of Program 12-2. You'll be prompted to set the lighting condition which will indicate that the switch is to turn on. Do this by exposing the analog light sensor to the amount of light at which you want the light to turn itself on. While the sensor is still exposed to this light level, press the X key on the computer. The computer stores this level of light as a number and will turn on the electric light when the sensor sees *less than* this amount of light.

The program also prompts you to set the *light off* condition. After the electric light has been turned on, when this second light level (or more light) is present, the electric light is turned off.

Using this circuit requires that the sensor be mounted near a window where it can detect outside light.

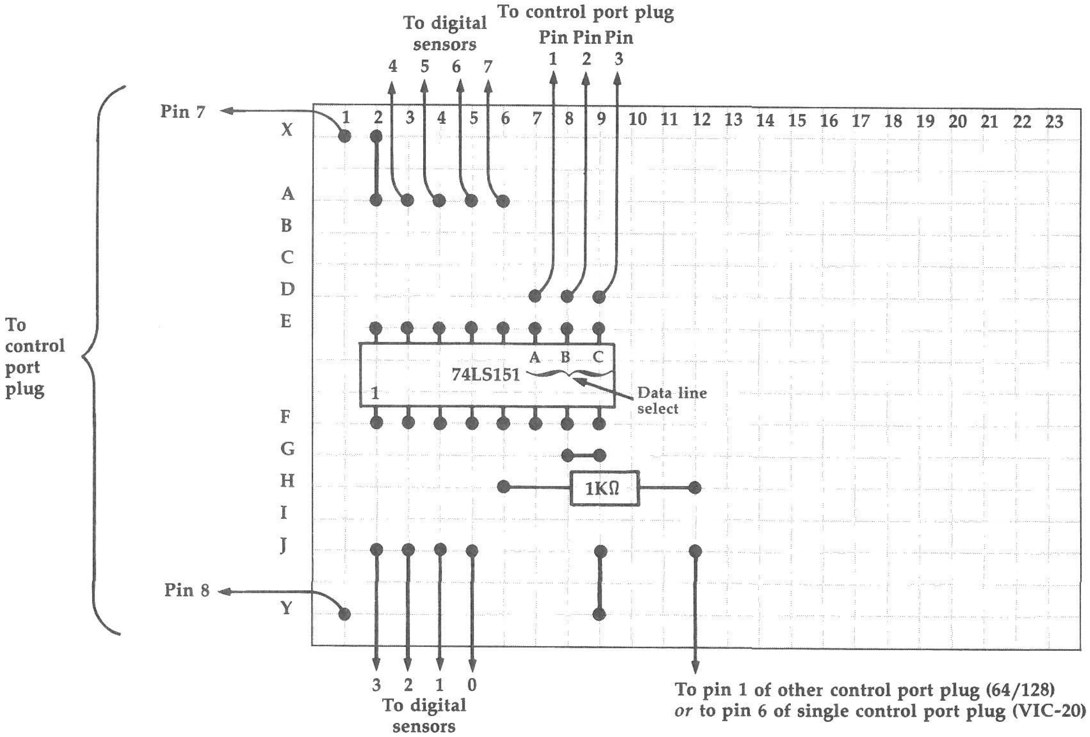
Whatever you do, don't mount the sensor beneath the electric light you're controlling.

Multiplexer Circuit

A *multiplexer* circuit lets your computer monitor many digital input signals using its limited number of I/O lines. To make an eight-input multiplexer, you'll have to have these components:

Quantity	**Part**	**Part Number**
2	9-pin D-subminiature female (only one necessary for VIC and Atari)	276-1538
1	74LS151 IC	None
1	1K ohm resistor 271-8023 (not required for Atari)	
1	Solderless breadboard	276-175

Figure 12-4. Solderless Breadboard Layout for a Multiplexer Circuit


To digital sensors
4 5 6 7
To control port plug
Pin Pin Pin
1 2 3
Pin 7
Pin 8
To control port plug
X
A
B
C
D
E
F
G
H
I
J
Y
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
74LS151
A B C
1
Data line select
1KΩ
3 2 1 0
To digital sensors
To pin 1 of other control port plug (64/128)
or to pin 6 of single control port plug (VIC-20)

The multiplexer circuit primarily consists of a number of input lines from various digital sensors.

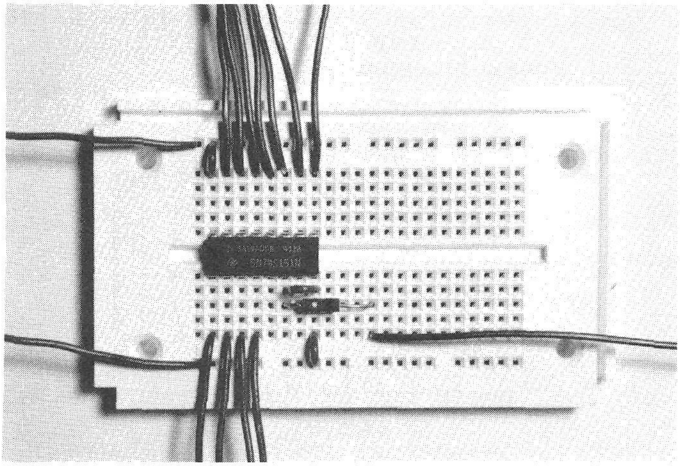
Follow these steps to put together your multiplexer:

1. Insert the 74LS151 IC so that its pin 1 goes into point F2 of the solderless breadboard, and its pin 9 into point E9.
2. Solder a 4-inch-long wire to pin 1 of one of the 9-pin plugs (for the Atari and VIC-20 versions, solder this wire to pin 6 of the single 9-pin plug required). This wire carries the selected data line's signal to the computer.
3. Connect the other end of this wire to point J12 of the breadboard.
4. The 1K ohm resistor bridges the points H12 to H6 of the board. (When building the Atari version, use a jumper wire in place of the resistor.)
5. Solder wires about four inches long to pins 1, 2, 3, 7, and 8 of the second 9-pin plug (in the case of the Atari and VIC versions, solder these wires to the same pins of the single 9-pin plug).
6. The wires from pins 7 and 8 provide the power to the circuit. Connect the wire from pin 7 to point X1, and the wire from pin 8 to location Y1 of the board.
7. The wires from pins 1, 2, and 3 insert into plug points D7, D8, and D9 respectively. These last three pins are used to select the data line signal "seen" by the computer.
8. Connect jumper wires as follows:

Jumper	**From**	**To**
1	Y9	J9
2	X2	A2
3	G8	G9

Pins 1–4 and 12–15 of the 74LS151 IC are the input lines of the multiplexer. They can be connected to any sensor which provides an on/off signal. The sensors may be electronic (like the digital light sensor) or simply switches between the pins and ground.

Figure 12-5. Multiplexer Still Life



A congregation of wires from the multiplexer lead off to various digital sensors.

If you've built a multiplexer for the Commodore 64 or Commodore 128, the 9-pin plug with one wire attached is inserted into control port 2. The other 9-pin connector plugs into control port 1.

Since you used only one 9-pin plug for the Atari version, insert it into control port 2.

Program 12-3 allows the circuit to monitor eight digital inputs. It outputs three digital signals to the multiplexer circuit, which tell it to route one of the input signals to a pin of the computer's I/O port. The program sequentially selects each of the eight input lines and checks its condition. (This is called *polling.*) If any of the eight inputs is at a logic low state, the computer prints out such a message on the screen.

Program 12-3—Commodore 64/128

```
PK 10 PRINT "PRESS X TO EXIT PROGRAM"
GM 20 REM SET DATA DIRECTION REGISTER FO
      R OUTPUT
QJ 30 POKE 56322,255
RP 40 FOR I=0 TO 7
```

```
PE 50 REM SET DATA OUTPUT
SR 60 POKE 56320,I
RG 70 IF (PEEK(56321)AND(2↑0))=0 THEN PR
      INT " SENSOR ";I
RK 80 NEXT I
EP 90 REM RESET DATA LINES FOR INPUT
RQ 100 POKE 56322,0
ES 110 GET B$
BX 120 IF B$="X" THEN END
CP 130 GOTO30
```

Program 12-3—VIC-20

```
CM 10 PRINT" MONITORING SENSORS"
HR 20 PRINT "PRESS X TO EXIT"
DP 30 REM SET DATA DIRECTION REGISTER FO
      R OUTPUT
QH 40 POKE 37139,(2↑2+2↑3+2↑4)
GM 50 FOR I=0 TO 7
SP 60 REM SET DATA OUTPUT
GH 70 POKE 37137,I*4
QC 80 IF (PEEK(37137)AND(2↑5))=0 THEN PR
      INT " SENSOR ";I
JM 90 NEXT I
GJ 100 POKE 37139,128
ES 110 GET B$
BX 120 IF B$="X" THEN END
MQ 130 GOTO40
GM 140 PRINT "TURN SWITCH OFF"
```

Program 12-3—Atari

```
ME 100 REM MULTIPLE INPUTS
JG 110 PRINT "MONITORING SENSORS"
ON 120 PRINT :PRINT "PRESS X TO EXIT
       "
FM 130 REM SET DATA DIRECTION REGIST
       ER FOR OUTPUT
FO 140 POKE 54018,48
CI 150 POKE 54016,7
FL 160 POKE 54018,52
DB 170 REM CYCLE THROUGH OUTPUT COMB
       INATION
BA 180 FOR I=0 TO 7
DO 190 POKE 54016,I
AF 195 FOR J=1 TO 10:NEXT J
EH 200 IF (STRIG(0)=0) THEN PRINT "S
       ENSOR ";I
```

```
LL 210 REM CHECK IF X KEY PRESSED
KB 220 IF PEEK(764)=22 THEN 250
BN 230 NEXT I
GM 235 GOTO 180
NG 240 REM CLEAR LAST KEYSTROKE
CP 250 POKE 764,255
AL 260 REM RESET DATA LINES FOR INPU
       T
GC 270 POKE 54018,48
CF 280 POKE 54016,0
FP 290 POKE 54018,52
```

A Demultiplexer Circuit

A *demultiplexing* circuit can be made using a 74LS154 IC. This chip takes four input signals from the computer to output a signal to one of 16 lines. The output line corresponding to the input signal combination goes logic low, while the other output lines of this IC remain high.

Quantity	**Part**	**Part Number**
1	9-pin D-subminiature female	276-1538
1	74LS154 IC	None
1	Solderless breadboard	276-175

You'll also need some solid copper wire for jumpers and some stranded copper wire for other connections.

1. Insert the 74LS154 IC so that its pin 1 goes into point G6 of the solderless breadboard, and its pin 24 into point C6.
2. Solder wires about four inches long to pins 1, 2, 3, and 4 of the 9-pin plug.

These wires connect to the breadboard at these locations:

Wire from Pin	To Point
1	A7
2	A8
3	A9
4	A10

3. Solder two wires to pins 7 and 8 of the 9-pin plug. Connect the wire from pin 7 to point X1, and the wire from pin 8 to location Y1 of the board.
4. Solder wires for the digital output lines to these points on the breadboard:

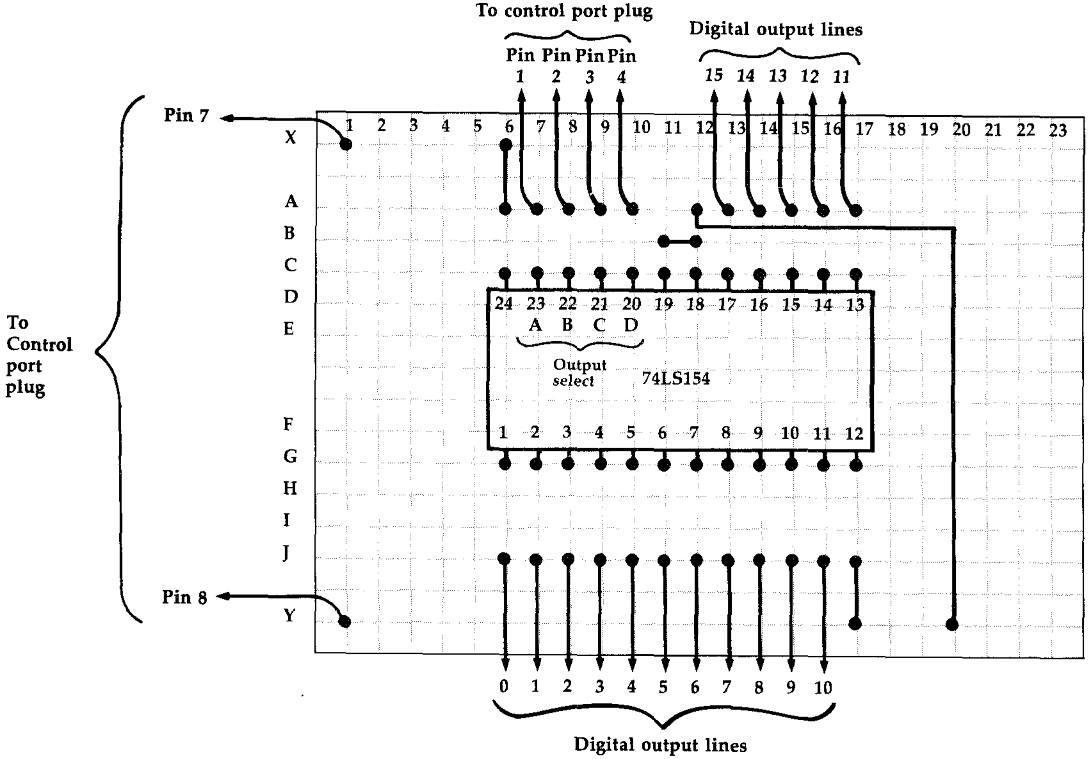
Digital Output Line	To Point
0	J6
1	J7
2	J8
3	J9
4	J10
5	J11
6	J12
7	J13

Digital Output Line	To Point
8	J14
9	J15
10	J16
11	A17
12	A16
13	A15
14	A14
15	A13

5. Connect jumper wires as follows:

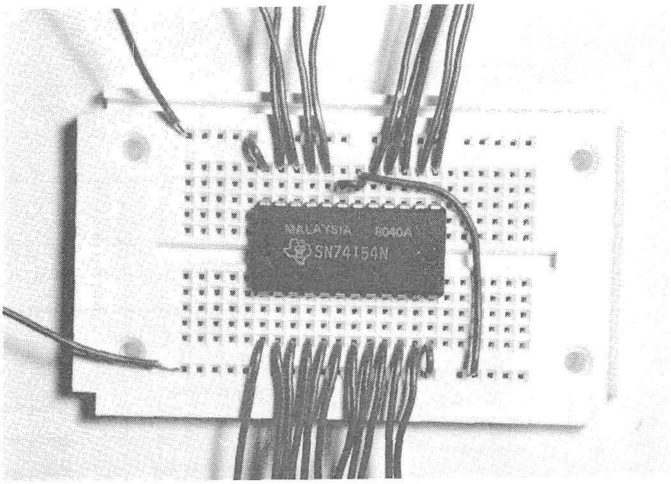
Jumper	From	To
1	Y17	J17
2	X6	A6
3	A12	Y20
4	B11	B12

Figure 12-6. Demultiplexer Circuit Solderless Breadboard Layout


To control port plug
Pin Pin Pin Pin
1 2 3 4
Digital output lines
15 14 13 12 11
Pin 7
Pin 8
To Control port plug
X
A
B
C
D
E
F
G
H
I
J
Y
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
24 23 22 21 20 19 18 17 16 15 14 13
A B C D
Output select
74LS154
1 2 3 4 5 6 7 8 9 10 11 12
0 1 2 3 4 5 6 7 8 9 10
Digital output lines

The demultiplexer's main component is a large, 24-pin integrated circuit.

Figure 12-7. More Wires Than You Can Count



Up to 16 output lines can be run from the demultiplexer circuit.

When you've finished, plug the 9-pin plug into control port 2.

The 74LS154 is interfaced to the computer using four lines from the control port. These four lines output the signals that select which of the output lines goes logic low. The outputs from the demultiplexer can be used to turn on electronic switches or other devices which require digital signals. The power for the demultiplexer circuit, as for the multiplexer, is obtained from pins 7 and 8 of the computer's control port.

Sending signals to set an output line of the demultiplexer is very similar to sending the output required from the computer to select an input line of the multiplexer. The only difference is that four output lines from the computer are used instead of three. Try writing a program which will sequentially set each of the demultiplexer's output lines to a logic low state. You can verify your program using your logic probe.

CHAPTER 13

Robotics

13 Robotics

One of the most exciting applications of digital electronics and computers is robotics. While a robotics control system is beyond the scope of this book, what you'll find here are some ideas to get you started.

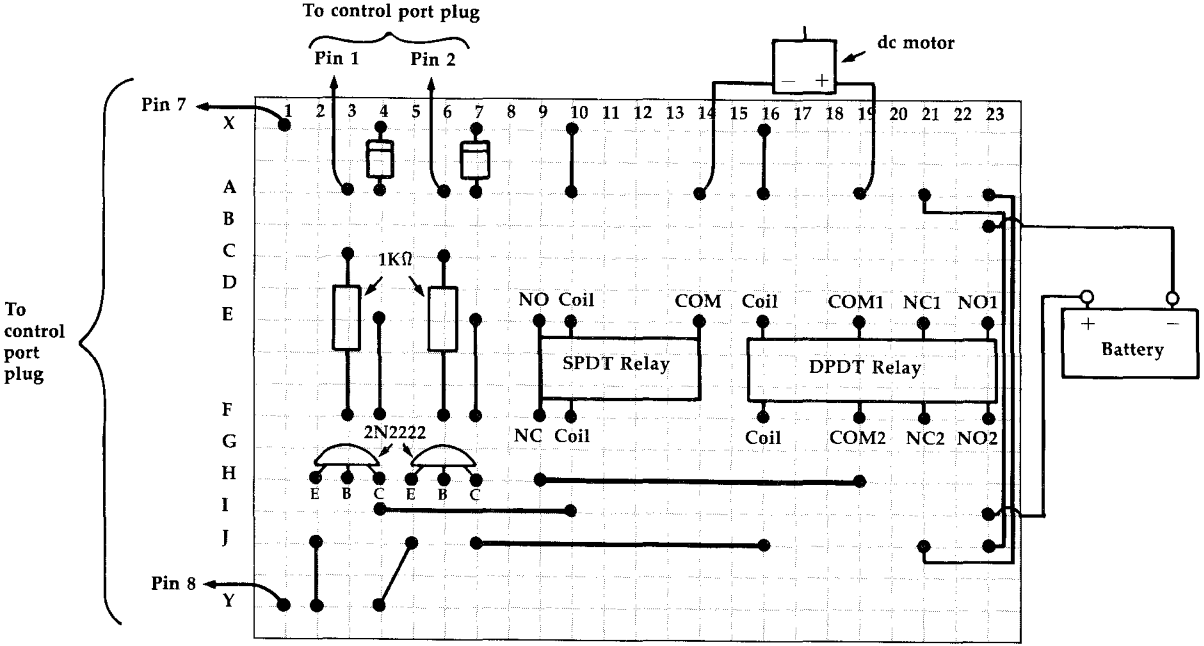
The actions of most robots are carried out by motors. To move, for instance, a mobile robot's wheels are turned; when a robot's arm goes up, the arm's joints are moved by motors. These motors, in turn, are controlled by a computer. The computer is programmed to turn the motors on and off so that the robot moves in a coordinated fashion to perform a task.

You already have enough expertise to control a dc motor with your computer. You can simply switch it on and off by using an electronic switch circuit. If you use a second electronic switch circuit, with a DPDT relay in place of the SPDT, you can control a dc motor to turn in both directions.

Though you've already built an electronic switch circuit in Chapter 8, it will be easier if you just begin from scratch when putting together a dc motor control circuit. Here's what you'll need:

Quantity	Part	Part Number
1	9-pin D-subminiature female	276-1538
2	2N2222 NPN transistors	276-1617
2	2.2K ohm resistors	271-8027
2	IN914 diodes	276-1620
1	5-volt SPDT DIP relay	275-243
1	DPDT DIP relay 5-volt coil	275-215
1	Solderless breadboard	276-175

Figure 13-1. DC Motor Breadboard Layout—Commodore Version


To control port plug
Pin 1
Pin 2
dc motor
Pin 7
X
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
A
B
C
D
E
F
G
H
I
J
Y
Pin 8
To control port plug
1KΩ
2N2222
E B C E B C
NO Coil
COM
SPDT Relay
NC Coil
Coil
COM1
NC1
NO1
DPDT Relay
COM2
NC2
NO2
+
−
Battery

Abbreviations

NO	Normally open terminal
NC	Normally closed terminal
COM	Common Terminal
Coil	Coil Terminal

This circuit, though considerably more complex, is much like that of the electronic switch you created in Chapter 8.

Quantity	Part	Part Number
1	7400 IC (Atari)	276-1801
1	DC battery-powered motor	None

You'll also need some solid copper wire for connections to the 9-pin plug, motor, and battery, as well as for jumpers on the solderless breadboard.

If you're building the Atari version of the control circuit, you'll need a longer than normal solderless breadboard.

The Commodore Version

Here's how to build the Commodore version of the dc motor control circuit.

1. Wire the 9-pin plug. Cut four pieces of wire, each about four inches long, and remove about 1/4 inch of the insulation from the wires' ends.
2. Solder wires to pins 1, 2, 7, and 8 of the 9-pin plug.
3. Connect the four wires as follows: The wire from pin 8 connects to point Y1 on the solderless breadboard. The wire from pin 7 connects to point X1 of the solderless breadboard. The last two lines, from pins 1 and 2, connect to locations A3 and A6 respectively.
4. Connect the two 2.2K ohm resistors between points C3 and F3, and points C6 and F6 on the solderless breadboard.
5. Mount one 2N2222 transistor so that its base inserts into point H3, its emitter into point H2, and its collector into point H3.
6. The second 2N2222 transistor should be mounted so that its base inserts into location H6, its emitter into point H7, and its collector into H5.

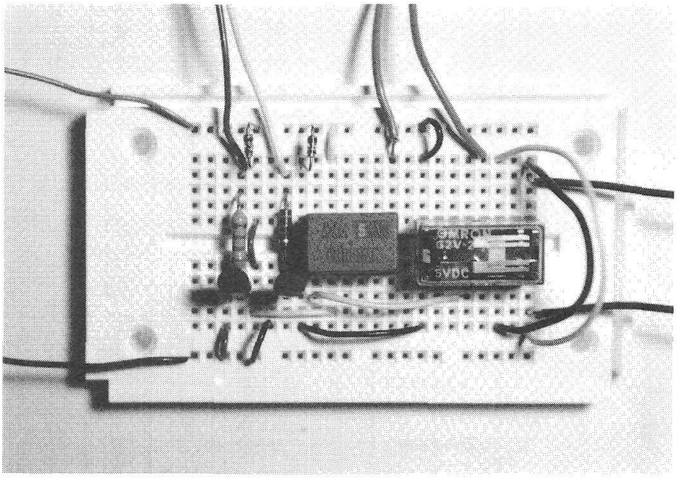
7. The first diode is connected so that its cathode lead inserts into point X4 of the solderless breadboard and its anode lead connects to point A4. (A band around one end of the diode is generally used to identify the cathode lead.)
8. The second diode should be placed so that its cathode lead goes into location X7, while its anode lead connects to point A7.
9. The SPDT relay is inserted so that its normally open and normally closed pins attach to points F9 and E9 respectively. The common pin connects to point E14. Its coil terminal should be connected to points E10 and F10.
10. Insert the DPDT relay so that its normally open pins 1 and 2 attach to points E23 and F23 respectively. Its normally closed pins 1 and 2 should be connected to points E21 and F21 respectively. The two common pins, 1 and 2, are attached to locations E19 and F19. Finally, the coil terminal is connected to points E16 and F16.
11. The following jumper connections are required:

Wire	**From**	**To**
1	X10	A10
2	X16	A16
3	E4	F4
4	E7	F7
5	A21	J23
6	A23	J21
7	H9	H19
8	I4	I10
9	J7	J16
10	J2	Y2
11	J5	Y4

12. Solder two wires to the battery you'll use, one wire to the positive pole, the other to the negative pole.

13. Connect the positive pole wire to point I23 of the solderless breadboard. Connect the negative pole wire to point B23.
14. Solder two wires to the dc motor, one to its positive pole, the other to its negative pole.
15. Connect the positive pole wire to point A19 of the breadboard and the negative pole wire to location A14.

Figure 13-2. The Finished Commodore Board



This solderless breadboard contains the most complex circuit project in this book.

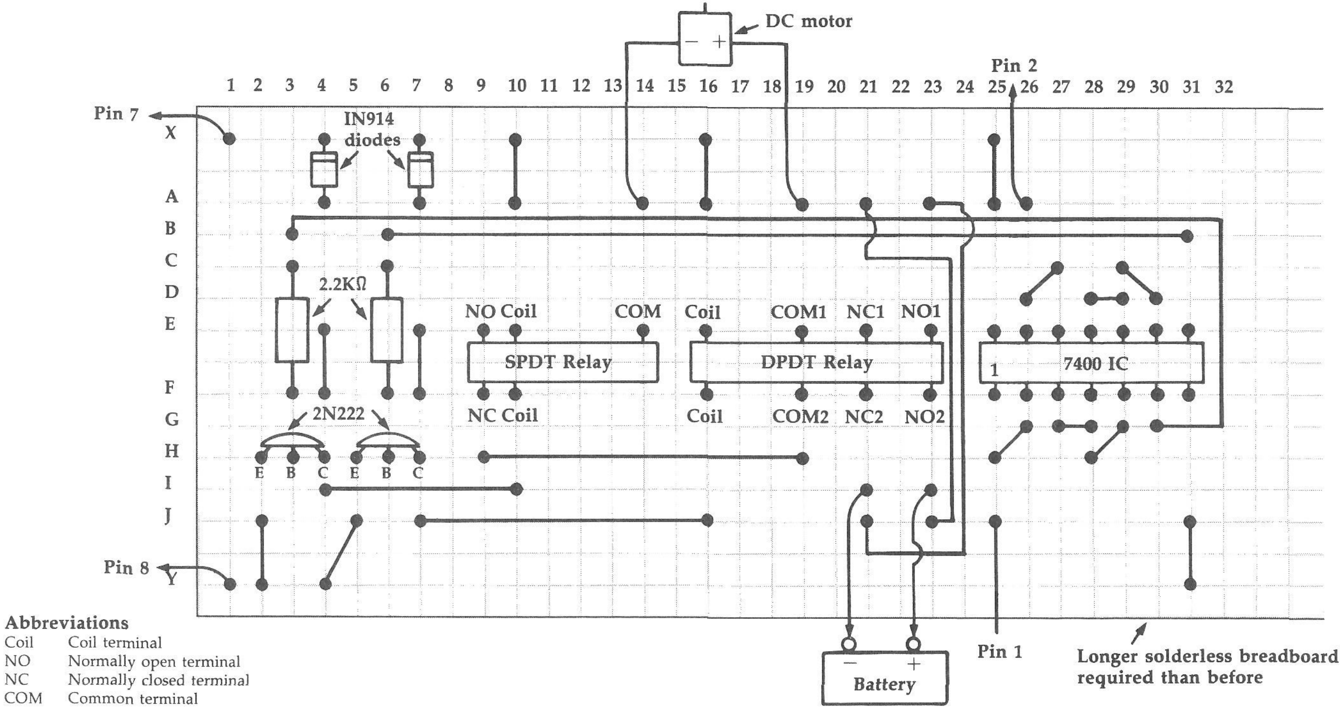
Changes When Constructing the Atari Version

Refer to Figure 13-3 for the layout of the Atari version.

Substitute the following steps in the Atari version:

3. The wires from pins 1 and 2 connect to points J25 and A26 respectively.

Figure 13-3. Solderless Breadboard Layout—Atari Version


DC motor
−
+
Pin 2
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32
Pin 7
X
A
B
C
D
E
F
G
H
I
J
Y
Pin 8
IN914 diodes
2.2KΩ
2N222
E B C E B C
NO Coil
COM
SPDT Relay
NC Coil
Coil
COM1
NC1
NO1
DPDT Relay
Coil
COM2
NC2
NO2
1
7400 IC
Battery
Pin 1
Longer solderless breadboard required than before

Abbreviations

Coil	Coil terminal
NO	Normally open terminal
NC	Normally closed terminal
COM	Common terminal

A 7400 IC component and several jumpers are just some of the differences between the Commodore and Atari versions of the dc motor control circuit.

11. These are the jumper connections for the Atari version:

Wire	**From**	**To**
1	X10	A10
2	X16	A16
3	E4	F4
4	E7	F7
5	B3	G30
6	B6	B31
7	D26	C27
8	D28	D29
9	C29	D30
10	H25	G26
11	G27	G28
12	H28	G29
13	A21	J23
14	A23	J21
15	J31	Y31
16	H9	H19
17	I4	I10
18	J7	J16
19	J2	Y2
20	J5	Y4

13. Connect the positive pole wire from the battery to point I23 of the solderless breadboard. Connect the negative pole wire to point I21.

16. Insert the 7400 IC into the solderless breadboard so that its pins 1 and 8 go into plug points F25 and E31 respectively.

Warning: If you use a larger motor, remember to mind the ratings of the solderless breadboard and its relays.

The circuit uses two signals from the computer—one to switch on the motor and the other to reverse the battery polarity to the motor. Reversing the battery polarity to a dc motor causes its shaft to rotate in the opposite direction.

The motor can be used to turn the wheels of a small mobile "turtle" robot. If you use a dc gear motor with a reasonably low rpm speed and enough torque, you can even control a joint of a robot, such as the elbow or shoulder of a mechanical arm. You can use several of these motor control circuits to operate a multijoint robot. Consider using a demultiplexer to send the digital signals to different motor control circuits.

Some robots simply turn on their motors for a certain amount of time, assuming that the parts manipulated by the motors reach particular points. But if, for example, something obstructs their movements, these robots continue working, oblivious to their situations. More intelligent robots use sensors to inform them about their environments. If you build a mobile robot, you can use microswitches to check whether your robot has bumped into a wall. If it requires several such sensors, you may have to use a multiplexing circuit. You can also mount potentiometers to turn with the joints of a mechanical-arm robot. The values returned by the A/D converter of your computer allow you to keep track of the arms' positions.

Once you build the motor control circuit, you can control it from your computer in much the same way as the following demonstration program does. Plug the circuit into control port 1 of your computer. (The Commodore 64/128 version of the program requires that you insert a joystick into control port 2.)

Program 13-1—Commodore 64/128

```
AK 100 REM MOTOR CONTROL DEMONSTRATION
SS 110 REM REQUIRES A JOYSTICK
HC 115 PRINT "JOYSTICK CONTROLS":PRINT
SA 120 PRINT"UP - MOTOR TURNS IN ONE DIR
       ECTION"
CH 130 PRINT"DOWN - MOTOR TURNS IN OPPOS
       ITE DIRECTION"
CE 132 PRINT:PRINT"PRESS FIRE BUTTON TO
       {SPACE}EXIT DEMO"
DB 135 B=0
HK 140 A=PEEK(56320)
CG 150 IF(A AND 2↑0)=0 THEN GOSUB 1000:
       GOTO 140
```

```
EP 160 IF (A AND 2↑1)=0 THEN GOSUB 2000:
       GOTO 140
CQ 170 GOSUB 3000
PM 180 IF (A AND 2↑4)=0 THEN END
JX 190 GOTO 140
CB 1000 REM TURN MOTOR ON IN ONE DIRECTI
        ON
CM 1010 REM CHECK IF THIS HAS ALREADY BE
        EN DONE
FK 1020 IF B=1 THEN 1100
MA 1030 REM SET FIRST DATA LINE FOR OUTP
        UT
SM 1040 POKE 56323,PEEK(56323)OR(2↑0)
GD 1050 REM SET FIRST DATA LINE LOGIC LO
        W TO TURN ON MOTOR
EG 1060 POKE 56321,PEEK(56321)ANDNOT(2↑0
        )
KD 1070 B=1
RF 1100 RETURN
JJ 2000 REM TURN ON MOTOR IN OTHER DIREC
        TION
KJ 2010 REM CHECK IF THIS HAS ALREADY BE
        EN DONE
KK 2020 IF B=2 THEN 2100
CR 2030 REM SET BOTH DATA LINES FOR OUTP
        UT
QX 2040 POKE 56323,PEEK(56323)OR(2↑0+2↑1
        )
CM 2050 REM SET BOTH DATA LINES LOGIC LO
        W TO TURN ON MOTOR AND REVERSE P
        OLARITY
MQ 2060 POKE 56321,PEEK(56321)ANDNOT(2↑0
        +2↑1)
RC 2070 B=2
MC 2100 RETURN
MC 3000 REM TURN MOTOR OFF
EC 3010 REM CHECK AND SEE IF THIS HAS BE
        EN DONE
JP 3020 IF B=0 THEN 3100
KM 3030 REM TURN BOTH DATA LINES LOGIC H
        IGH
BM 3040 POKE 56321,PEEK(56321)OR(2↑0+2↑1
        )
SG 3050 REM RESET BOTH DATA LINES FOR IN
        PUT
PX 3060 POKE 56323,PEEK(56323)ANDNOT(2↑0
        +2↑1)
XS 3070 B=0
HA 3100 RETURN
```

Program 13-1—VIC-20

```
XS 10 PRINT "KEYBOARD CONTROLS":PRINT
SB 20 PRINT"< - MOTOR TURNS IN ONE DIREC
      TION"
KR 30 PRINT"> - MOTOR TURNS IN OPPOSITE
      {SPACE}DIRECTION"
DE 40 PRINT:PRINT"PRESS SPACE BAR TO EXI
      T DEMO"
CH 50 B=0
EH 60 A=PEEK(197)
MS 70 IF A=29 THEN GOSUB120:GOTO60
PS 80 IF A=37 THEN GOSUB210:GOTO60
MK 90 GOSUB300
XD 100 IF A=32 THEN POKE 198,0:END
CP 110 GOTO60
FJ 120 REM TURN MOTOR ON IN ONE DIRECTIO
       N
PP 130 REM CHECK IF THIS HAS ALREADY BEE
       N DONE
SE 140 IF B=1 THEN200
XK 150 REM SET FIRST DATA LINE FOR OUTPU
       T
FJ 160 POKE 37139,PEEK(37139)OR(2↑2)
CS 170 REM SET FIRST DATA LINE LOGIC LOW
        TO TURN ON MOTOR
RF 180 POKE 37137,PEEK(37137)ANDNOT(2↑2)
FF 190 B=1
MC 200 RETURN
JQ 210 REM TURN ON MOTOR IN OTHER DIRECT
       ION
QR 220 REM CHECK IF THIS HAS ALREADY BEE
       N DONE
QC 230 IF B=2 THEN290
PR 240 REM SET BOTH DATA LINES FOR OUTPU
       T
PR 250 POKE 37139,PEEK(37139)OR(2↑2+2↑3)
PX 260 REM SET BOTH DATA LINES LOGIC LOW
       TO TURN ON MOTOR AND REVERSE POL
       ARITY
AG 270 POKE 37137,PEEK(37137)ANDNOT(2↑2+
       2↑3)
MM 280 B=2
SK 290 RETURN
RP 300 REM TURN MOTOR OFF
CR 310 REM CHECK AND SEE IF THIS HAS BEE
       N DONE
EG 320 IF B=0 THEN380
BM 330 REM TURN BOTH DATA LINES LOGIC HI
       GH
```

```
QQ 340 POKE 37137,PEEK(37137)OR(2↑2+2↑3)
FH 350 REM RESET BOTH DATA LINES FOR INP
       UT
CQ 360 POKE 37139,PEEK(37139)ANDNOT(2↑2+
       2↑3)
KX 370 B=0
HS 380 RETURN
```

Program 13-1—Atari

```
AO 100 REM MOTOR CONTROL DEMONSTRATI
       ON
NF 110 PRINT "KEYBOARD CONTROL"
CA 120 PRINT
BF 130 PRINT "< - MOTOR TURNS IN ONE
        DIRECTION"
KJ 140 PRINT "> - MOTOR TURNS IN OPP
       OSITE DIRECTION"
LO 150 PRINT "S - STOP MOTOR"
AK 155 PRINT :PRINT "X - EXIT PROGRA
       M"
EG 160 B=0
CN 170 A=PEEK(764)
IK 180 IF A=54 THEN GOSUB 1000:GOTO
       170
IN 190 IF A=55 THEN GOSUB 2000:GOTO
       170
NF 200 GOSUB 3000
AK 210 IF A=22 THEN END
GF 220 GOTO 170
DG 1000 REM TURN MOTOR ON IN ONE DIR
        ECTION
OJ 1010 REM CHECK IF THIS HAS ALREAD
        Y BEEN DONE
PD 1020 IF B=1 THEN 1100
AB 1030 REM SET DATA LINES FOR OUTPU
        T
IP 1032 POKE 54018,48
FG 1034 POKE 54016,3
IO 1036 POKE 54018,52
NJ 1040 REM SET FIRST DATA LINE LOGI
        C LOW TO TURN ON MOTOR
FD 1050 POKE 54016,2
HH 1060 B=1
KC 1100 RETURN
NH 2000 REM TURN ON MOTOR IN OTHER D
        IRECTION
OK 2010 REM CHECK IF THIS HAS ALREAD
        Y BEEN DONE
```

```
PG 2020  IF B=2 THEN 2100
CP 2030  REM SET BOTH DATA LINES FOR
         OUTPUT
JA 2032  POKE 54018,48
FH 2034  POKE 54016,3
IP 2036  POKE 54018,52
DF 2040  REM SET BOTH DATA LINES LOGI
         C LOW TO TURN ON MOTOR AND R
         EVERSE POLARITY
FC 2050  POKE 54016,0
HJ 2060  B=2
KD 2100  RETURN
FM 3000  REM TURN MOTOR OFF
JJ 3010  REM CHECK AND SEE IF THIS HA
         S BEEN DONE
PG 3020  IF B=0 THEN 3100
JD 3030  REM SET BOTH DATA LINE LOGIC
          HIGH
FF 3040  POKE 54016,3
DL 3050  REM RESET DATA LINES FOR INP
         UT
JC 3060  POKE 54018,48
FF 3070  POKE 54016,0
JA 3090  POKE 54018,52
IA 3095  B=0
KE 3100  RETURN
```

Appendices

A

COMPUTE!'s Guide to Typing In Programs

Computers are precise—type each program in this book *exactly* as it's listed, including the necessary punctuation and symbols, except for special characters noted below. We've provided a special listing convention as well as a program to check your typing—"The Automatic Proofreader."

Programs for Commodore and Atari 800/XL/XE computers may contain some hard-to-read special characters, so we have a listing system which indicates these control characters. You'll find these Commodore and Atari characters in curly braces; *do not type the braces.* For example, {CLEAR} or {CLR} instructs you to insert the symbol which clears the screen on the Atari or Commodore machine. A complete list of these symbols is found in the table below. For Commodore and Atari computers, a single symbol by itself within curly braces is usually a control key or graphics key. If you see {A}, hold down the CONTROL key and press A. This will produce a reverse video character on the Commodore (in quote mode) or a graphics character on the Atari.

Graphics characters entered with the Commodore logo key are enclosed in a special bracket: [<A>]. In this case, you would hold down the Commodore logo key as you type A. Our Commodore listings are in uppercase, so shifted symbols are underlined. A graphics heart symbol (SHIFT-S) would be listed as S. One exception is {SHIFT–SPACE}. When you see this, hold down SHIFT and press the space bar. If a number precedes a symbol, such as {5 RIGHT}, {6 S}, or [<8 Q>], you would enter five cursor rights, six shifted S's, or eight Commodore-Q's. On the

Atari, inverse characters (white on black) should be entered with the inverse video key (Atari logo key on 400/800 models).

When You Read:	Press:	See:
{CLR}	SHIFT CLR/HOME	
{HOME}	CLR/HOME	
{UP}	SHIFT ↑ CRSR ↓	
{DOWN}	↑ CRSR ↓	
{LEFT}	SHIFT ← CRSR →	
{RIGHT}	← CRSR →	
{RVS}	CTRL 9	
{OFF}	CTRL 0	
{BLK}	CTRL 1	
{WHT}	CTRL 2	
{RED}	CTRL 3	
{CYN}	CTRL 4	
{PUR}	CTRL 5	
{GRN}	CTRL 6	
{BLU}	CTRL 7	
{YEL}	CTRL 8	

When You Read:	Press:	See:
{1}	COMMODORE 1	
{2}	COMMODORE 2	
{3}	COMMODORE 3	
{4}	COMMODORE 4	
{5}	COMMODORE 5	
{6}	COMMODORE 6	
{7}	COMMODORE 7	
{8}	COMMODORE 8	
{ F1 }	f1	
{ F2 }	SHIFT f1	
{ F3 }	f3	
{ F4 }	SHIFT f3	
{ F5 }	f5	
{ F6 }	SHIFT f5	
{ F7 }	f7	
{ F8 }	SHIFT f7	
←	←	

Whenever more than two spaces appear in a row, they are listed in a special format. For example, {6 SPACES} means press the space bar six times. Our Commodore listings never leave a single space at the end of a line, instead moving it to the next printed line as {SPACE}.

The Automatic Proofreader

Type in the appropriate program listed below, then save it for future use. The Commodore Proofreader works on the Commodore 128, 64, and VIC-20. Don't omit any lines, even if they contain unfamiliar commands or you think they don't apply to your computer. When you run the program, it installs a machine language program in memory and erases its BASIC portion automatically (so be sure to save several copies before running the program for the first time). If you're using a Commodore 128, do *not* use any GRAPHIC commands while the Proofreader is active. You should disable the Commodore Proofreader before running any other program. To do this, either turn the computer off and on or enter SYS 64738 (for the 64), SYS 65341 (128), or SYS 64802 (VIC-20). To reenable the Proofreader, reload the program and run it as usual. Unlike the original VIC/64 Proofreader, this version works the same with disk or tape.

On the Atari, run the Proofreader to activate it (the Proofreader remains active in memory as a machine language program); the BASIC loader automatically erases itself. Pressing SYSTEM RESET deactivates the Atari Proofreader; enter PRINT USR(1536) to reenable it.

Once the Proofreader is active, try typing in a line. As soon as you press RETURN, a pair of letters appears. The pair of letters is called a *checksum.*

Compare the value displayed on the screen by the Proofreader with the checksum printed in the program listing in the book. The checksum is given to the left of each line number. Just type in the program a line at a time (without the printed checksum), press RETURN or Enter, and compare the checksums. If they match, go on to the next line. If they don't, check your typing; you've made a mistake. Because of the checksum method used, do not type abbreviations, such as ? for PRINT. On the Atari Proofreader, spaces are not counted as part of the checksum, so be sure you type the right number of spaces between quotation marks. The Atari Proofreader does not check to see that you've typed the characters in the right order, so if characters are transposed, the checksum still matches the listing. The Commodore Proofreader catches transposition errors and ignores spaces unless they're enclosed in quotation marks.

Program 1. Commodore Proofreader

```
10 VEC=PEEK(772)+256*PEEK(773):LO=43:HI=44
20 PRINT"{CLR}{WHT}AUTOMATIC PROOFREADER FOR ";:IF
   VEC=42364 THEN PRINT "C-64"
30 IF VEC=50556 THEN PRINT "VIC-20{BLU}"
40 IF VEC=35158 THEN WAIT CLR:PRINT "PLUS/4 & 16"
50 IF VEC=17165 THEN LO=45:HI=46:WAIT CLR:PRINT"12
   8{WHT}"
60 SA=(PEEK(LO)+256*PEEK(HI))+6:ADR=SA
70 FOR J=0 TO 166:READ BYT:POKE ADR,BYT:ADR=ADR+1:
   CHK=CHK+BYT:NEXT
80 IF CHK<>20570 THEN PRINT "*ERROR* CHECK TYPING
   {SPACE}IN DATA STATEMENTS":END
90 FOR J=1 TO 5:READ RF,LF,HF:RS=SA+RF:HB=INT(RS/2
   56):LB=RS-(256*HB)
100 CHK=CHK+RF+LF+HF:POKE SA+LF,LB:POKE SA+HF,HB:N
   EXT
110 IF CHK<>22054 THEN PRINT "*ERROR* RELOAD PROGR
   AM AND CHECK FINAL LINE":END
120 POKE SA+149,PEEK(772):POKE SA+150,PEEK(773)
130 IF VEC=17165 THEN POKE SA+14,22:POKE SA+18,23:
   POKESA+29,224:POKESA+139,224
140 PRINT CHR$(147);CHR$(17);"PROOFREADER ACTIVE":
   SYS SA
150 POKE HI,PEEK(HI)+1:POKE (PEEK(LO)+256*PEEK(HI)
   )-1,0:NEW
160 DATA 120,169,73,141,4,3,169,3,141,5,3
170 DATA 88,96,165,20,133,167,165,21,133,168,169
180 DATA 0,141,0,255,162,31,181,199,157,227,3
190 DATA 202,16,248,169,19,32,210,255,169,18,32
200 DATA 210,255,160,0,132,180,132,176,136,230,180
210 DATA 200,185,0,2,240,46,201,34,208,8,72
220 DATA 165,176,73,255,133,176,104,72,201,32,208
230 DATA 7,165,176,208,3,104,208,226,104,166,180
240 DATA 24,165,167,121,0,2,133,167,165,168,105
250 DATA 0,133,168,202,208,239,240,202,165,167,69
260 DATA 168,72,41,15,168,185,211,3,32,210,255
```

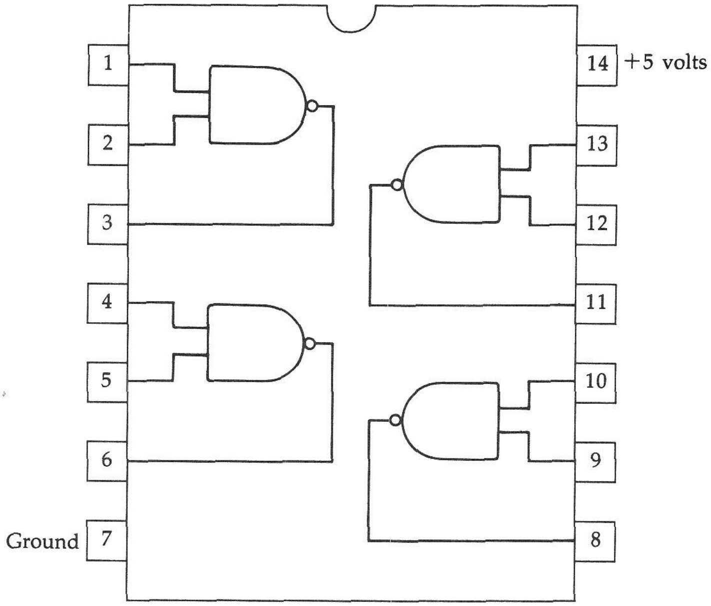
```
270 DATA 104,74,74,74,74,168,185,211,3,32,210
280 DATA 255,162,31,189,227,3,149,199,202,16,248
290 DATA 169,146,32,210,255,76,86,137,65,66,67
300 DATA 68,69,70,71,72,74,75,77,80,81,82,83,88
310 DATA 13,2,7,167,31,32,151,116,117,151,128,129,
   167,136,137
```

Program 2. Atari Proofreader

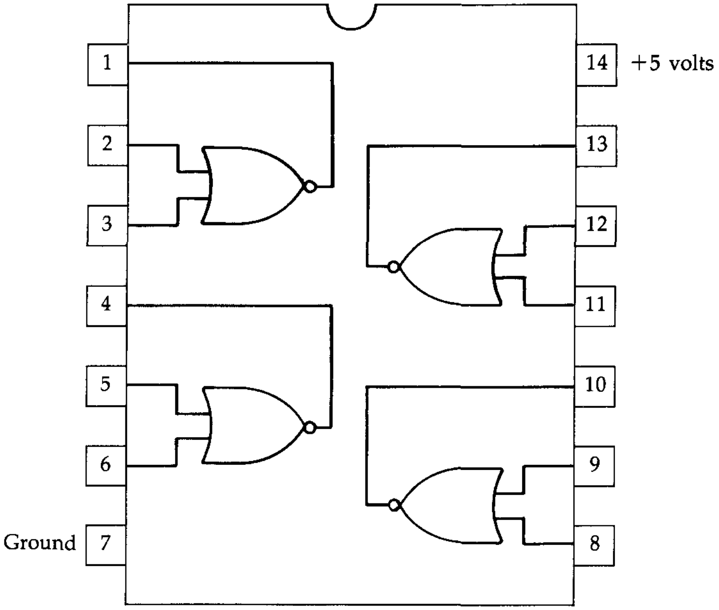
```
100 GRAPHICS 0
110 FOR I=1536 TO 1700:READ A:POKE I,A:CK=CK
    +A:NEXT I
120 IF CK<>19072 THEN ? "Error in DATA State
    ments.  Check Typing.":END
130 A=USR(1536)
140 ? :? "Automatic Proofreader Now Activate
    d."
150 NEW
1536 DATA 104,160,0,185,26,3
1542 DATA 201,69,240,7,200,200
1548 DATA 192,34,208,243,96,200
1554 DATA 169,74,153,26,3,200
1560 DATA 169,6,153,26,3,162
1566 DATA 0,189,0,228,157,74
1572 DATA 6,232,224,16,208,245
1578 DATA 169,93,141,78,6,169
1584 DATA 6,141,79,6,24,173
1590 DATA 4,228,105,1,141,95
1596 DATA 6,173,5,228,105,0
1602 DATA 141,96,6,169,0,133
1608 DATA 203,96,247,238,125,241
1614 DATA 93,6,244,241,115,241
1620 DATA 124,241,76,205,238,0
1626 DATA 0,0,0,0,32,62
1632 DATA 246,8,201,155,240,13
1638 DATA 201,32,240,7,72,24
1644 DATA 101,203,133,203,104,40
1650 DATA 96,72,152,72,138,72
1656 DATA 160,0,169,128,145,88
1662 DATA 200,192,40,208,249,165
1668 DATA 203,74,74,74,74,24
1674 DATA 105,161,160,3,145,88
1680 DATA 165,203,41,15,24,105
1686 DATA 161,200,145,88,169,0
1692 DATA 133,203,104,170,104,168
1698 DATA 104,40,96
```

B Integrated Circuits

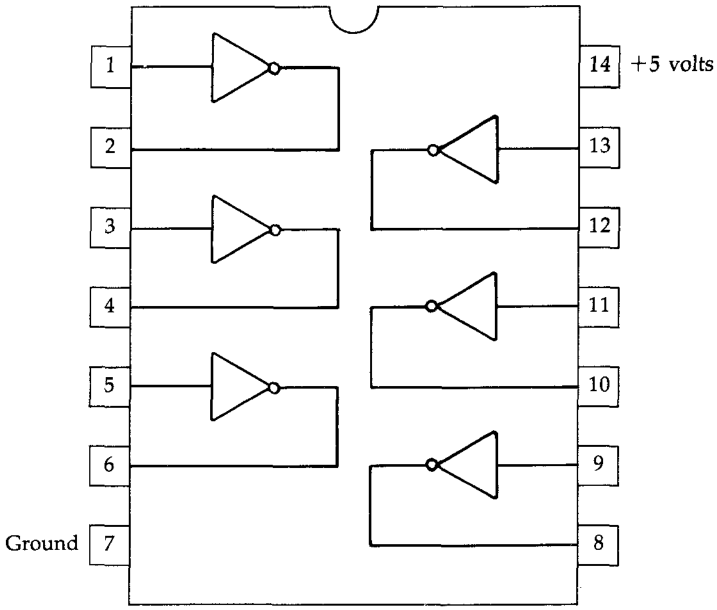
7400/74LS00
Quad NAND Gate


1
2
3
4
5
6
Ground
7
14
+5 volts
13
12
11
10
9
8

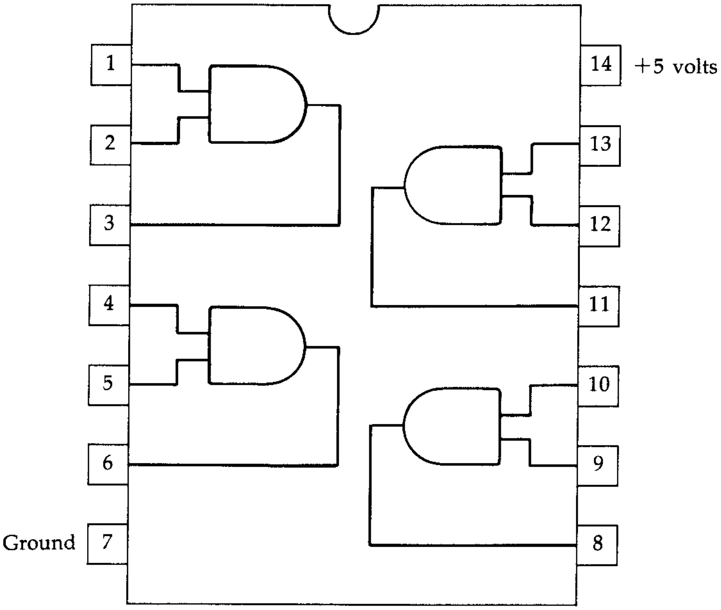
7402/74LS02
Quad NOR Gate


1
2
3
4
5
6
Ground
7
14
+5 volts
13
12
11
10
9
8

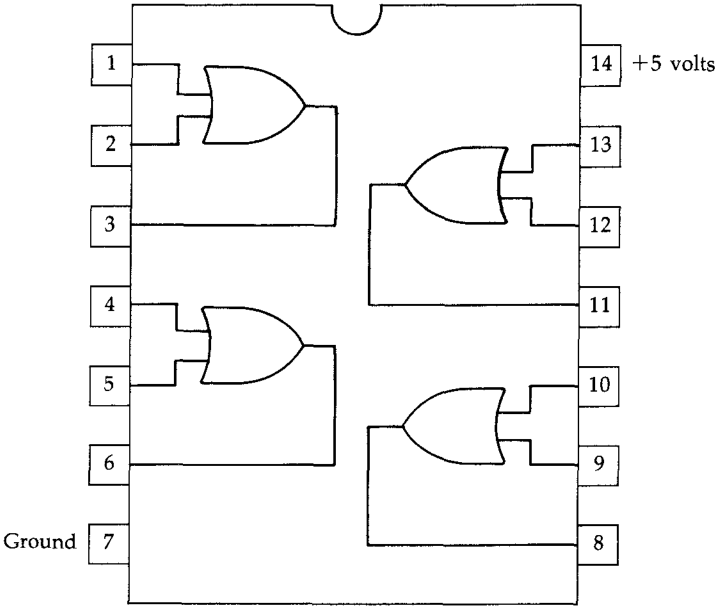
7404/74LS04
Hex Inverter


1
2
3
4
5
6
Ground
7
14
+5 volts
13
12
11
10
9
8

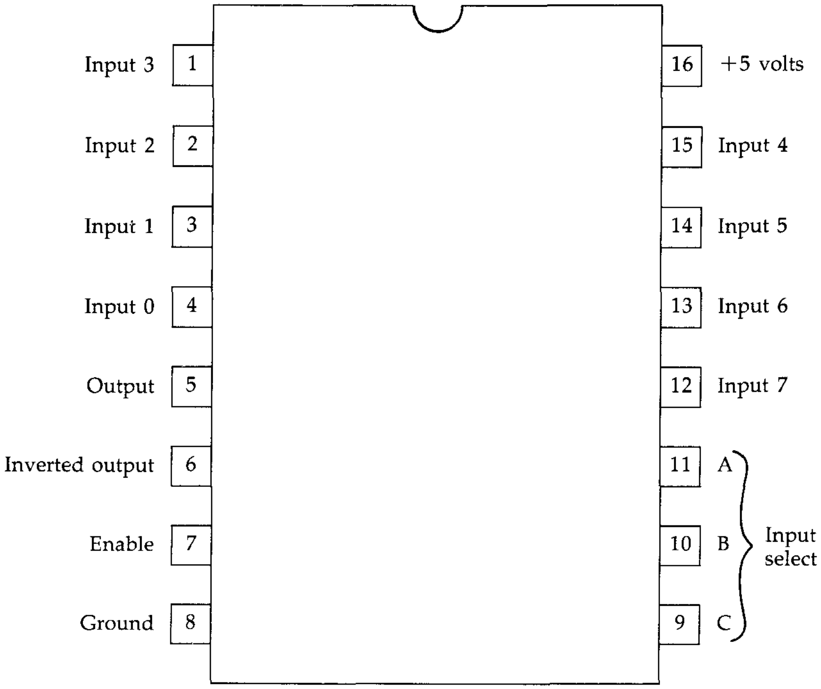
7408/74LS08
Quad AND Gate


1
2
3
4
5
6
Ground
7
14
+5 volts
13
12
11
10
9
8

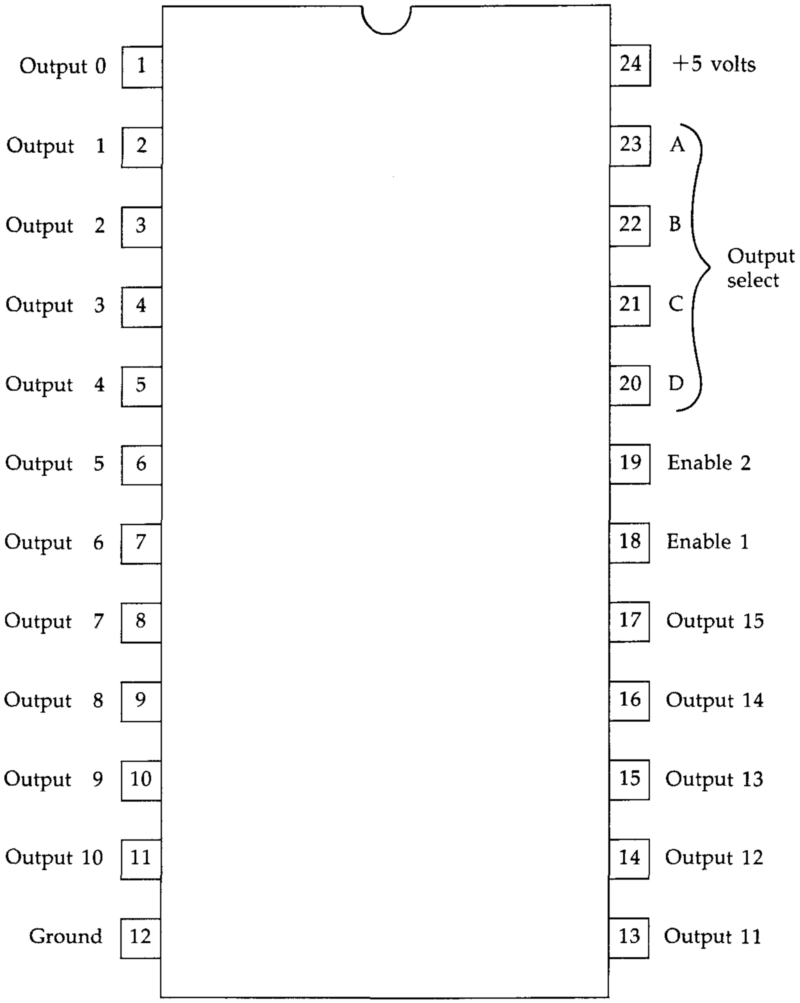
7432/74LS32
Quad OR Gate


1
2
3
4
5
6
Ground 7
14 +5 volts
13
12
11
10
9
8

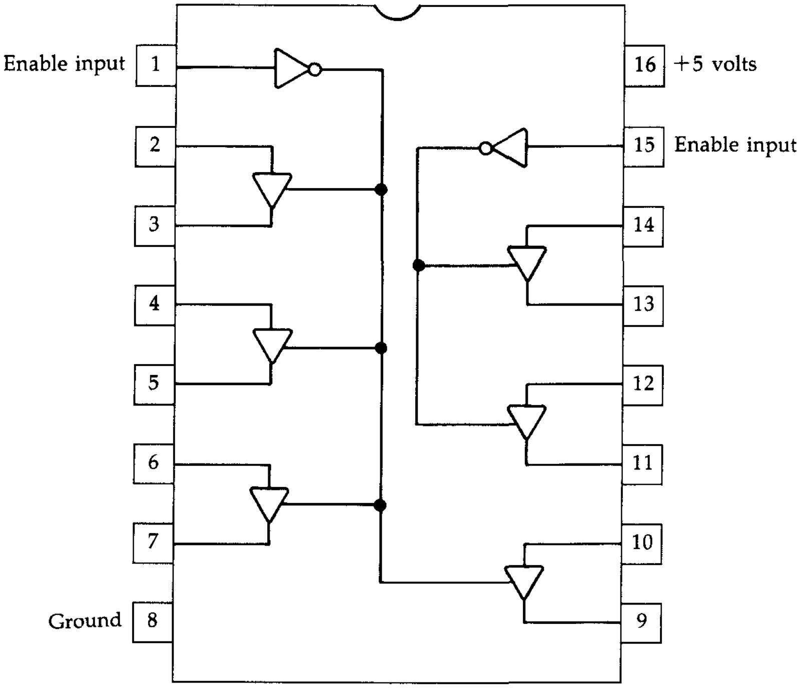
74151/74LS151
Eight Line to One Line Mutliplexer


Input 3
1
Input 2
2
Input 1
3
Input 0
4
Output
5
Inverted output
6
Enable
7
Ground
8
16
+5 volts
15
Input 4
14
Input 5
13
Input 6
12
Input 7
11
A
10
B
9
C
Input select

74154/74LS154
Four Line to 16 Line Demultiplexer


Output 0
1
Output 1
2
Output 2
3
Output 3
4
Output 4
5
Output 5
6
Output 6
7
Output 7
8
Output 8
9
Output 9
10
Output 10
11
Ground
12
24
+5 volts
23
A
22
B
21
C
20
D
Output select
19
Enable 2
18
Enable 1
17
Output 15
16
Output 14
15
Output 13
14
Output 12
13
Output 11

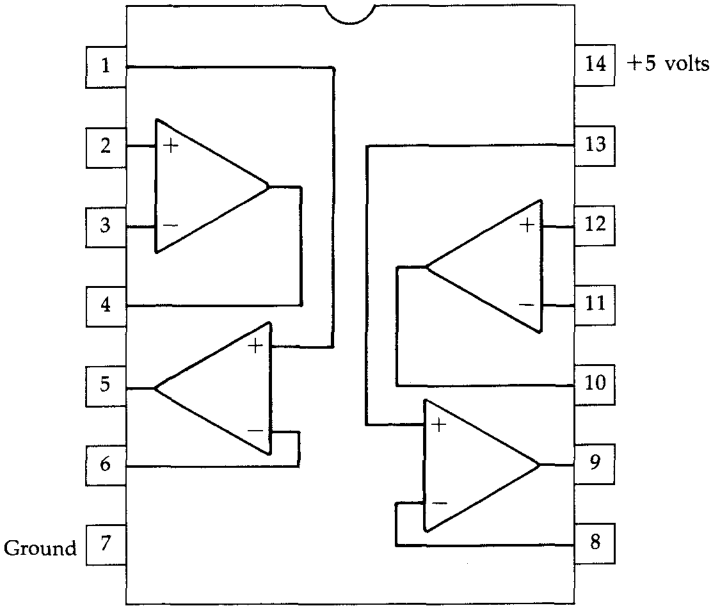
74367/74LS367
Hex Three-State Buffers


Enable input
1
2
3
4
5
6
7
Ground
8
16
+5 volts
15
Enable input
14
13
12
11
10
9

Truth Table

Enable Input	Input	Output
H	H or L	High impedance
L	L	L
L	H	H

3900
Quad Operational Amplifier


1
2
3
4
5
6
7
Ground
14
+5 volts
13
12
11
10
9
8
+
−
+
−
+
−
+
−

Index

actuators *vii* 97
A/D converter 33
analog circuit 3
analog devices 33
analog-to-digital converter. *See* A/D converter
applications *vii*
Atari BASIC 83
"Automatic Proofreader, The" 166–69
binary number system 3–6
bit 5
Boolean algebra 109
breadboarding *viii*. *See also* wiring, methods of
burglar alarm 97–106
  Atari program 105
  Commodore program 102
  constructing 99–100
  VIC-20 program 103
byte 6
circles, small 110
circuit, analog 3
circuit, building a *viii*
circuit, digital 3
cold joint *xii*
control port pin 82, 109
data *vii*
data direction register 81, 82
  Atari 82–83
  Commodore 81–82
data port 81
dc motor control circuit 151
  Atari program 161–62
  Commodore program 158–59
  constructing (Atari version) 155–57
  constructing (Commodore version) 153–55
  VIC-20 program 160–61
decoders 115–18
demultiplexer 115–18, 158
demultiplexer circuit 145–48
  constructing 145–46
digital circuitry 109
digital light beam timer 74–78
  Atari program 78
  Commodore program 76
  VIC-20 program 77
digital light sensor 97, 98, 129
  variable 67
digital logic 109–20
digital signal 67, 109
  high 67
  low 67
D-subminiature female connectors 9
electromagnet 90
electromechanical switch 90
electron beam 59–61
electron gun 53
electronic signals 81
electronic switch 81–94, 98, 136
  Atari program 93
  Commodore program 92
  constructing 84–88
  using 91, 94
  VIC-20 program 92
electrons 53
encoders 115–18
game paddles 33–41
  Atari program 39
  Commodore program 38
  constructing 34–35
  "ML Paddle Reader" program 41
  reading 39–40
  VIC-20 program 39
gates 109–15
  AND 111
  combining 143
  exclusive NOR 113
  exclusive OR 112
  NAND 111
  noninverting buffer 110
  NOR 112
  NOT 110
  OR 112
H (logic high) 110
IC chip. *See* integrated circuit chip
icons *xiii*
input 18
input/output ports 9
integrated circuit chip *xi*, 118
  pins 54
integrated circuits 171–79
  demultiplexer, four line to 16 line 177
  hex inverter 173
  hex three-state buffers 178
  multiplexer, eight line to one line 176
  quad AND gate 174
  quad NAND gate 171
  quad NOR gate 172
  quad operational amplifier 179
  quad OR gate 175
interrupt request (IRQ) 39
inverters 110
I/O pin 82
I/O ports 9

IRQ. *See* interrupt request
joystick 97
  constructing 23–29
  gravity 28–29
  port 9–11
  push button 23–27
  push button, constructing 24–27
joystick port connector 16–20
  Atari program 19
  Commodore program 18
  constructing 16–17
  VIC-20 program 19
L (logic low) 100
light-emitting diode (LED) 12, 91
light pen 53–64, 67
  Atari program 64
  Commodore program 62
  constructing 54–58
  programming with 61–62
  VIC-20 program 63
light sensor, analog 45–50, 136
  Atari program 49
  Commodore program 49
  constructing (Atari version) 47–48
  constructing (Commodore version) 45–47
light sensor, digital 67–78
  Atari program 73
  Commodore program 73
  constructing 68–71
  VIC-20 program 73
light switch 136–40
  Atari program 139
  Commodore program 137
  VIC-20 program 138
location 83
logic probe 10–13, 123–26
  constructing 10–13, 123–25
  using 126
logic states
  high 15
  low 15, 20
low-power Schotkey (LS) 120
"ML Paddle Reader" program 41
monitor 53
multiplexer circuit 140–45, 158
  Atari program 144
  Commodore program 143
  constructing 142
  VIC-20 program
multiplexers 115–18
multiplier 13
NAND gates 60
noninverting buffer 60, 90
number systems 4–6
  binary 3–6
  decimal 4
ohms 13
output 18
output signal 109
PADDLE(X) 34, 50
phototransistor 60, 72
polling 143
ports 9, 10
  data 81
  input/output 9
  joystick 9–11
potentiometer 33
power ratings 14
registers 81
resistor color codes 14
resistors 13, 14, 33, 126
resistor variable 45
robotics 151–62
sensors *vii,* 81, 97, 109
software *vii*
soldering 55
soldering iron, heating *ix–xii*
STICK 18, 74, 82
STRIG 18
switch 3, 72
  Atari version 169
  Commodore version 168
  tilt 28
switch sensor 98, 100
  infrared 98
  magnetic switch 98
  metallic tape 98
  ultrasonic 98
  vibration detectors 98
tilt switches 28
timer circuit 74
tin *xii*
tolerance 13
transistor-transistor logic signal level. *See* TTL
truth table 109
TTL 3, 120
two-beam digital timer circuit 129–36
  Atari program 135
  Commodore program 134
  constructing 130
  using 134
  VIC-20 program 135
typing in programs 165–69
voltage 3, 33
watts 14
wiring, methods of *viii*
  circuit boards *viii*
  mounting *viii*
  plug points *ix*
  wire wrapping *viii*

COMPUTE! Books

Ask your retailer for these **COMPUTE! Books** or order directly from **COMPUTE!**.

Call toll free (in US) **800-346-6767** (in NY 212-887-8525) or write COMPUTE! Books, P.O. Box 5038, F.D.R. Station, New York, NY 10150.

Quantity	Title	Price*	Total
	SpeedScript: The Word Processor for the Commodore 64 and VIC-20 (94-9)	$ 9.95	
	Commodore SpeedScript Book Disk	$12.95	
	128 Machine Language for Beginners (033-5)	$16.95	
	COMPUTE!'s First Book of the Commodore 128 (059-9)	$14.95	
	COMPUTE!'s Commodore 64/128 Collection (97-3)	$12.95	
	All About the Commodore 64, Volume Two (45-0)	$16.95	
	All About the Commodore 64, Volume One (40-X)	$12.95	
	Programming the Commodore 64: The Definitive Guide (50-7)	$24.95	
	COMPUTE!'s Data File Handler for the Commodore 64 (86-8)	$12.95	
	COMPUTE!'s Kids and the Commodore 128 (032)	$14.95	
	Kids and the Commodore 64 (77-9)	$12.95	
	COMPUTE!'s Commodore Collection, Volume 1 (55-8)	$12.95	
	COMPUTE!'s Commodore Collection, Volume 2 (70-1)	$12.95	
	COMPUTE!'s Personal Accounting Manager for the Commodore 64 (014-9)	$12.95	
	COMPUTE!'s VIC-20 and Commodore 64 Tool Kit: BASIC (32-9)	$16.95	
	COMPUTE!'s VIC-20 and Commodore 64 Tool Kit: Kernal (33-7)	$16.95	
	COMPUTE!'s Telecomputing on the Commodore 64 (009)	$12.95	
	COMPUTE!'s VIC-20 Collection (007)	$12.95	
	Programming the VIC (52-3)	$24.95	
	VIC Games for Kids (35-3)	$12.95	
	Mapping the VIC (24-8)	$14.95	

*Add $2.00 per book for shipping and handling.
Outside US add $5.00 air mail or $2.00 surface mail.

NC residents add 4.5% sales tax ____________
Shipping & handling: $2.00/book ____________
Total payment ____________

All orders must be prepaid (check, charge, or money order).
All payments must be in US funds.
☐ Payment enclosed.
Charge ☐ Visa ☐ MasterCard ☐ American Express

Acct. No.____________________ Exp. Date__________
(Required)
Name____________________
Address____________________
City____________________ State __________ Zip__________

*Allow 4–5 weeks for delivery.
Prices and availability subject to change.
Current catalog available upon request.

COMPUTE! Books

P.O. Box 5038
F.D.R. Station
New York, NY 10150

Ask your retailer for these **COMPUTE! Books**. If he or she has sold out, order directly from **COMPUTE!**.

For Fastest Service
Call Our **TOLL FREE US Order Line**
1-800-346-6767
In NY call 212-265-8360
Or write **COMPUTE! Books**,
P.O. Box 5038, F.D.R. Station, New York, NY 10150

Quantity	Title	Price	Total
______	COMPUTE!'s First Book of Atari (00-0)	**$12.95**	______
______	COMPUTE!'s Second Book of Atari (06-X)	**$12.95**	______
______	COMPUTE!'s Third Book of Atari (18-3)	**$12.95**	______
______	COMPUTE!'s First Book of Atari Graphics (08-6)	**$12.95**	______
______	COMPUTE!'s Second Book of Atari Graphics (28-0)	**$12.95**	______
______	COMPUTE!'s First Book of Atari Games (14-0)	**$12.95**	______
______	Inside Atari DOS (02-7)	**$19.95**	______
______	COMPUTE!'s Atari Collection, Volume 1 (79-5)	**$12.95**	______
______	COMPUTE!'s Atari Collection, Volume 2 (029-7)	**$14.95**	______
______	Machine Language for Beginners (11-6)	**$14.95**	______
______	Second Book of Machine Language (53-1)	**$14.95**	______
______	*SpeedScript:* The Word Processor for the Atari (003)	**$ 9.95**	______
______	Mapping The Atari, Revised (004)	**$16.95**	______
______	COMPUTE!'s ST Programmer's Guide (023-8)	**$16.95**	______
______	The Elementary Atari ST (024)	**$16.95**	______
______	COMPUTE!'s Kids and the ST (386)	**$14.95**	______
______	Elementary ST BASIC (343)	**$14.95**	______
______	Introduction to Sound and Graphics on the Atari ST (035)	**$14.95**	______

Add $2.00 per book shipping and handling. Outside US add $5.00 air mail or $2.00 surface mail.

NC residents add 4.5% sales tax ______

Shipping & handling ______

Total payment ______

All orders must be prepaid (money order, check, or charge). All payments must be in US funds.

☐ Payment enclosed Please charge my: ☐ Visa ☐ MasterCard
☐ American Express

Acct. No. ______________________ Exp. Date ______

Name ______________________________

Address ______________________________

City ________________ State ________ Zip ______

Country ______________________________

*Allow 4–5 weeks for delivery.
Prices and availability subject to change without notice.

If you've enjoyed the articles in this book, you'll find the same style and quality in every monthly issue of **COMPUTE!** Magazine. Use this form to order your subscription to **COMPUTE!**.

For Fastest Service
Call Our **Toll-Free** US Order Line
1-800-247-5470
In IA call 1-800-532-1272

COMPUTE!
P.O. Box 10954
Des Moines, IA 50340

My computer is:
☐ Commodore 64 or 128 ☐ TI-99/4A ☐ IBM PC or PCjr ☐ VIC-20
☐ Apple ☐ Atari ☐ Amiga ☐ Other ____________________
☐ Don't yet have one...

☐ $24 One Year US Subscription
☐ $45 Two Year US Subscription
☐ $65 Three Year US Subscription

Subscription rates outside the US:
☐ $30 Canada and Foreign Surface Mail
☐ $65 Foreign Air Delivery

Name ____________________
Address ____________________
City ____________ State ________ Zip ________
Country ____________________

Payment must be in US funds drawn on a US bank, international money order, or charge card.
☐ Payment Enclosed ☐ Visa
☐ MasterCard ☐ American Express

Acct. No. ____________________ Expires ____ / ____
(Required)

Your subscription will begin with the next available issue. Please allow 4–6 weeks for delivery of first issue. Subscription prices subject to change at any time.

Now Your Computer Can Control the Outside World

With a little work, a few electronic components, and some expert guidance, you can turn your personal computer into anything from a light detector to a burglar alarm.

Electronic Computer Projects shows you, step by step, how to build a wide variety of devices from inexpensive, easily obtained parts, and how to connect them to your Commodore VIC-20, 64, or 128, or Atari 800, XL, or XE computer. Complete with photographs, detailed figures, and schematics, this book is understandable—no jargon or undecipherable instructions—so that even a beginner can master each project.

Here's some of what you'll learn how to build with *Electronic Computer Projects:*

- A burglar alarm to safeguard your home or office.
- Sensors and detectors that can turn lights on at dusk, off at dawn.
- Push button joysticks for your computer.
- A light pen for graphics programs or menu selections.
- Game paddles for fast-moving arcade games.
- Logic probes, both simple and advanced, for "looking" at what's going on in a digital circuit.
- And more, from multiplexers to motor controllers.

Each project's parts are clearly listed, each step is clearly described. Ready-to-type-in programs are included for most projects, letting you test out each device and adjust it if necessary.

Electronic Computer Projects is a book which does more than *tell* you about computers and their possibilities—it *shows* you just what your machine can do.

ISBN 0-87455-052-1